**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»**

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

"На правах рукопису"
УДК_____

«До захисту допущено»
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
"___"_____2018р.

# Магістерська дисертація

зі спеціальності 121 Інженерія програмного забезпечення
за спеціалізацією Програмне забезпечення розподілених систем
на тему " iOS додаток управління педагогічними та науковими аспектами роботи кафедри "

Виконав: студент 6 курсу, групи **ТР-71мп**
Касьяненко Ігор Ігорович _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник к.т.н, доцент Карпенко Є. Ю. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018

**Національний технічний університет України**
**“Київський політехнічний інститут імені Ігоря Сікорського”**

Факультет теплоенергетичний
(повна назва)

Кафедра автоматизації проектування енергетичних процесів і систем
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
«___»_____20___ р.

**ЗАВДАННЯ**
**на магістерську дисертацію студенту**

Касьяненко Ігорю Ігоровичу
(прізвище, ім’я, по батькові)

**1. Тема дисертації:** iOS додаток управління педагогічними та науковими аспектами роботи кафедри

**науковий керівник дисертації** Карпенко Евген Юрійович, к.т.н.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « » 2018 р. № _____

**2. Термін подання студентом дисертації:** 10 грудня 2018 року

**3. Об’єкт дослідження:** Засоби розробки iOS додатків

**4. Предмет дослідження:** Засоби управління педагогічними та науковими аспектами роботи кафедри

**5. Перелік питань, які потрібно розробити:**

5.1. Аналіз сучасних програмних комплексів управління педагогічними та науковими аспектами роботи кафедри.
5.2. Аналіз програмних середовищ, що придатні для розробки у iOS.
5.3 Аналіз шляхів оптимізації iOS додатків.
5.4. Розробка оптимізованих елементів iOS додатків.
5.5. Тестування розробленого додатоку на результативність прискорення.
5.6. Розробка охоронної системи, що використовує розроблений додаток.
5.7. Розробка стартап-проекту.

**6. Орієнтовний перелік ілюстративного матеріалу:**

6.1. Презентація PowerPoint відповідно до теми дисертації.

**7. Дата видачі завдання** «__11__»_____вересня_____2018р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів виконання магістерської дисертації | Термін виконання етапів магістерської дисертації | Примітка |
|---|---|---|---|
| | Затвердження теми роботи | 17.05.2018 | |
| | Вивчення та аналіз задачі. Проведення дослідження по вибраній темі | 01.06.2018-03.09.2018 | |
| | Розробка архітектури та загальної структури системи | 03.09.2018-28.09.2018 | |
| | Програмна реалізація системи | 01.10.2018-26.10.2018 | |
| | Захист програмного продукту | 22.10.2018 | |
| | Оформлення пояснювальної записки | 02.09.2018-10.12.2018 | |
| | Передзахист | 26.11.2018-30.11.2018 | |
| | Захист | 19.12.2018 | |

Студент _____ Касьяненко І.І._____
( підпис ) (прізвище та ініціали)

Науковий керівник _____ Карпенко Е.Ю._____
( підпис ) (прізвище та ініціали)

# РЕФЕРАТ

**Структура й обсяг дипломної роботи.**

Магістерська дисертація складається зі вступу, п'яти розділів, висновку, переліку посилань з N найменувань, N додатки, і містить N рисунки, N таблиці. Повний обсяг магістерської дисертації складає N сторінок, з яких перелік посилань займає 7 сторінок, додатки – 7 сторінок.

**Актуальність теми.** В роботі розглянуто дві теми: автоматизація частини освітнього процесу та дослідження підходів до розробки мобільного ПЗ для створення більш ефективного та гнучкого продукту. Ці теми є актуальними, бо:

— в освітній сфері не вистачає сучасних електронних рішень, які б автоматизували певні аспекти роботи;

— методології розробки постійно оновлюються, програмування зводиться до все більш абстрактного рівня.

**Мета дослідження** полягає у визначенні способів підвищення якості додатків за рахунок удосконалення існуючих методів та засобів розробки.

Для досягнення поставленої задачі були сформульовані наступні завдання дослідження, що визначили логіку дослідження та його структуру:

— проаналізувати існуючі засоби розробки під iOS (нативні та кросплатформні);

— проаналізувати існуючі універсальні методи, патерни та принципи розробки програмного забезпечення;

— удосконалити підхід до розробки додатків під iOS з урахуванням особливостей платформи;

— розробити програмний продукт для автоматизації одного з бізнес-процесів на кафедрі з урахуванням досліджених та удосконалених методів розробки.

**Об’єктом дослідження** є методи, патерни та принципи розробки а також платформа iOS.

**Предметом** дослідження є нові засоби розробки під iOS, що спиратимуться та будуть удосконалювати існуючі засоби.

**Методи дослідження.** Розв'язання поставлених задач виконувались засобами комп'ютерного програмування та вивчення існуючих напрацювань програмістів, зокрема були вивчені методи розробки:

— нативне програмування із використанням патернів MVC, MVVM, VIPER, MVVMR;

— кросплатформне програмування.

**Наукова новизна одержаних результатів.** Найбільш суттєвими науковими результатами магістерської дисертації є:

— удосконалено спосіб проектування архітектури додатків під платформу iOS, що збільшило надійність та гнучкість розробки мобільних застосунків;

— розроблено продукт, що може бути використано для автоматизації певного аспекту освітнього процесу, який не було автоматизовано вже існуючими додатками.

**Практичне значення** одержаних результатів роботи полягає в розробці програмних рішень, які спростять розробку мобільних додатків на ОС iOS а також розробка додатку що автоматизує один з робочих процесів на кафедрі.

**Ключові слова.** *ПАТЕРН, АРХІТЕКТУРА, МОБІЛЬНИЙ ДОДАТОК, IOS, НАТИВНИЙ, КРОСПЛАТФОРМНИЙ.*

# ABSTACT

**The structure and volume of the thesis.**

Master's thesis consists of an introduction, five chapters, conclusion, list of references with N titles, N annexes, and contains N figures, N tables. The full range of master's thesis is N pages with a list of links takes N pages, apps - N pages.

**Topicality of the theme.** Two topics were raised in the diploma: automation of the educational process at the department and researching approaches to developing mobile software to create a more efficient and flexible product. Both themes are relevant, since:

— the educational sphere lacks in modern electronic solutions that would automate certain aspects of work;

— development methodologies are constantly updated, programming is reduced to an increasingly abstract level.

**The purpose and problems of research** is to determine the ways to improve the quality of applications by improving existing methods and tools.

To accomplish the task, the following research objectives were formulated, which determined the logic of the research and its structure:

— analyze existing iOS development tools (native and cross-platform);

— analyze existing universal methods, patterns and principles of software development;

— improve the approach to developing applications for iOS, taking into account the features of the platform;

— develop a software product for automating one of the business processes at the department, taking into account the research and advanced methods of development.

**The object of the research** is the methods, patterns and principles of development, as well as the platform iOS.

**The subject of the research** is the new iOS development tools that will build on and improve the existing tools.

**The solution** of the set tasks was performed by means of computer programming and studying of existing developments of programmers community, in particular, the methods of development were studied:

— native programming using MVC, MVVM, VIPER, MVVMR patterns;

— cross-platform programming.

**Scientific novelty of the results.** The most significant scientific results of the master's thesis are:

— improved the way to design an application architecture for the iOS platform, which has increased the reliability and flexibility of developing mobile applications;

— Developed product can be used to automate a particular aspect of the educational process, which has not been automated with existing applications which has been developed.

**The practical value** of the results of the work is to develop software that will facilitate the development of mobile applications on the iOS platform, as well as the development of an application that automates one of the work processes at the department.

**Keywords.** PATTERN, ARCHITECTURE, MOBILE APPLICATION, IOS, NATIVE, CROSSPLATFORM.

# ЗМІСТ

# ВСТУП

Розвиток комп‘ютерів і мережі інтернет створює основу для стрімкого поширення розподілених мобільних та веб додатків. Багато тривіальних розподілених систем застарівають та заміняються більш сучасними та ефективними електронними аналогами.

Особливу увагу слід звернути на системи, які забезпечують навчальний процес.

В розвинених європейських країнах питання модернізації освіти ставиться пріоритетною ціллю.

В Україні проблеми освіти розглядаються недостатньо. Нормою є застарілі програми, методології навчання, майже відсутнє впровадження електронних технологій в освіту, це все призводить до загальної регресії освітньої системи.

Однією з насущних проблем вищих навчальних закладів є повільні та малоефективні схеми взаємодії студентів з викладачами, відсутність уніфікованого інструменту впливу на процес навчання через інтернет, застарілість організації процесу вибору тем дипломних робіт студентів та їх узгодження.

Не завжди узгоджений графік прийому та перевірки лабораторних робіт, нечіткі терміни та правила спричиняють створення черг студентів, розпливчате усвідомлення процесу написання та здачі завдань, це все спричиняє неефективний розподіл часу та зусиль студентів та викладачів.

Роздрібненість інформації про кафедри, їх направлення, досягнення створює розпливчасте усвідомлення загального стану факультету, тим самим, відштовхуючи абітурієнтів.

Завдання полягає у створенні розподіленої системи, яка дозволить зручно та ефективно організувати процес взаємодії викладачів зі студентами, надасть консолідовану інформацію про напрями та роботу кафедр, дозволить вирішувати організаційні моменти в онлайн режимі.

Звіт включає шість розділів. Перший розділ описує постановку задачі та проблеми, які треба вирішити. У другому розділі проводиться огляд існуючих рішень,

їхніх переваг та недоліків, та порівняння з рішенням, що розроблюється. У третьому розділі розглядаються технології побудови клієнтських та серверних додатків, можливі архітектури та описуються інструменти розробки. У четвертому розділі наведено опис програмної реалізації: структуру програмного продукту та модулі програми. П’ятий розділ розкриває методику роботи користувача з програмною системою та графічний інтерфейс користувача. Шостий розділ описує продукт у якості стартап-проекту.

# 1 ЗАДАЧА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ДОДАТКУ ДЛЯ УПРАВЛІННЯ ПЕДАГОГІЧНИМИ ТА НАУКОВИМИ АСПЕКТАМИ КАФЕДРИ

Задача розробки сервісу є досить широкою та має нюанси, які розробник повинен дослідити[1], аби забезпечити максимальну комфортність інтерфейсу та ефективне використання ресурсів. Для побудови архітектури додатку необхідно визначитись із сутностями та юзкейсами даної предметної області. Для виділення сутностей та юзкейсів системи необхідно проаналізувати вже існуючі подібні рішення (для втілення подібного функціоналу) та виділити унікальний функціонал проектованої системи.

Метою є розробка мобільного додатку для управління педагогічними та науковими аспектами кафедри.

Для досягнення поставленої мети необхідно:

— провести аналіз предметної області;

— виділити сутності предметної області;

— виділити функціонал системи;

— реалізувати обрані функції;

— проаналізувати вже існуючі модулі та програмні продукти, що відповідають функціям майбутньої системи;

— обрати засоби розробки системи;

— реалізувати всі функції системи, які поставленні замовником.

Програмне забезпечення повинно мати наступні функції:

— реєстрація та авторизація;

— управління профілем (перегляд, редагування);

— перегляд інформації про наукові лабораторії;

— перегляд інформації про викладачів;

— управління дипломними роботами (створення, запропонування, відстеження, редагування);

— захист приватних даних користувача системи[2].

Даний сервіс складається з двох модулів — сервер для зберігання та обробки даних користувачів і мобільний додаток, що дозволяє оперувати даними, що знаходяться на сервері (створювати, редагувати та видаляти дані, проводити пошук та відтворення через користувацький інтерфейс).

Вхідними даними є дії користувача, що направлені на взаємодію із сутностями дипломних робіт та користувачів (створення, пошук, перегляд). Дані зберігаються на віддаленому сервері у реляційній базі даних. Зв'язки між даними в таблицях встановлюються в серверному додатку, параметри для сервера будуються та відправляється з клієнтського додатку

Вихідна інформація — сутності, що було створено на віддаленому сервері, запит і відображення яких відбуваються у клієнтському додатку.

Потенційні користувачі розроблюваного програмного забезпечення — студенти та викладачі кафедри АПЕПС.

## Висновки до розділу 1

У розділі було висвітлено мету магістерської дисертації, також визначені кроки для реалізації мети та оголошені вимоги до розроблюваної програми, коротко описані модулі програми, розподілення сервісу на клієнтську та серверну частини, визначені поняття вхідних та вихідних даних а також потенційні користувачі програмного комплексу.

# 2. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ ДЛЯ ОРГАНІЗАЦІЇ НАУКОВОЇ РОБОТИ КАФЕДРИ

Автоматизація освітньої системи проходить поряд з іншими галузями. Створені додатки частково або повністю замінюють тривіальні методи організації наукової роботи. В Україні подібні сервіси не мають широкого поширення.

## 2.1 Аналіз існуючих програмних засобів

Основною проблемою існуючих сервісів є орієнтація на веб платформу та фокусування на конкретному вузі чи кафедрі. Мобільні додатки для організації наукової роботи відсутні.

Одним з сервісів є Campus KPI – веб-сервіс для студентів та викладачів КПІ. Даний сервіс має широкий функціонал, що частково перетинається з функціоналом даного програмного продукту.

Кампус надає можливість поширювати навчальні матеріали (методичок, книжок, завдань лабораторних робіт та інше), також є модуль оцінювання виконаних робіт онлайн, може працювати як новинний портал університету.

Даний сервіс виконує більш конкретну функцію, дозволяє організувати процес ведення дипломної роботи починаючи від створення теми роботи та визначення керівника до узгодження графіку виконання роботи.

І викладачі, і студенти є власниками смартфонів. Мобільні пристрої знаходяться поряд зі своїм власником протягом доби довше ніж більшість інших гаджетів, таких як десктоп чи ноутбук, мають системи повідомлень(push-нотифікації) це робить їх ефективним способом оперативного обміну інформацією[3].

Для зв’язку з викладачами через інтернет студенти зазвичай використовують електронну пошту, інколи вони звертаються до соціальних мереж чи месенджерів. В

двох випадках є проблеми. Електронна пошта перевіряється не часто, мінусом є також неструктурованість тем листів, що надходять, це може спричиняти загублення важливих повідомлень серед безлічі інших, соціальні мережі слугують більш для неформального та приватного листування, аніж для вирішення робочих моментів.

З'являються такі спеціалізовані сервіси як Кампус чи розроблюваний програмний продукт.

Кампус має невисоку частоту відвідування, сервіс неоптимізовано для використання з портативних пристроїв, веб-платформа накладає обмеження на інформування про події та зміни. Таким чином, цей продукт не відповідає критерію оперативного надання інформації.

На відміну від кампуса, даний продукт орієнтований на мобільні телефони і має систему сповіщень, таки чином і студент, і викладач завжди мають актуальну інформацію щодо статусів дипломних робіт, тобто плюсом даної системи є автоматизованість. Студент і викладач завжди мають змогу подивитися ситуацію інших викладачів, відпадає необхідність питань щодо завантаженості студентами.

## 2.2. Порівняльний аналіз нативної та кросплатформної розробки

Розробка мобільного додатку на платформі iOS може проводиться за допомогою нативних [4] або кросплатформних [5] засобів, в обох підходах можна виділити переваги й недоліки.

Нативними є інструменти, які використовують стандартні блоки та API, визначені для певної системи компанією - розробником. Так, наприклад, мова Swift [6] є нативною для iOS, а Javascript в контексті Cordova [7] — ні.

Було визначено, що до переваг нативних додатків відноситься:

— швидкодія та оптимізоване використання ресурсв додатком;

— повноцінний UI та UX [8];

— краща індексованість додатку в AppStore

— велике коммьюніті розробників та різноманітність бібліотек.

Кросплатформними є засоби, які написані за допомогою одної мови чи SDK але можуть працювати на девайсах з різними операційними системами.

Після огляду всіх існуючих кросплатформних рішень було виділено наступні переваги:

— швидкість написання з нуля. Більшість юз-кейс класів може бути використано для декількох платформ одразу;

— економічна ефективність. Розробка одного додатка для двох платформ дешевша за розробку двох нативних рішень;

— випуск оновлень. Легше синхронізувати версії додатку, підтримувати єдність поведінки на всіх платформах.

Як відомо, продукти, які призначені для виконання більш вузького функціоналу працюють ефективніше аніж ті, що охоплюють більш широкі функції. Окрім зазначених переваг, кросплатформні рішення мають вагомі недоліки:

— проблеми із швидкодією. В основному кросплатформні додатки працюють через проміжний-шар, наприклад, вбудований веб-браузер, або проміжна мова, це накладає загальне пониження швидкодії

— проблеми з UX. Кожна платформа має власні особливості побудови інтерфейсів і принципів роботи користувача з ними. Нестандартні елементи управління та поведінка може смутити користувача та службу перевірки додатків в AppStore.

— нестабільність та застарілість деяких кросплатформних рішень. Більшість кросплатформних засобів розробляються компаніями що не є розробниками нативного SDK, і як результат, часто не встигають реалізувати функціонал подібний до того що виходить у світі нативної розробки;

— повнота та якість документації.

— обмежений доступ до системних сервісів(Bluetooth [9], WiFi та ін.)

Після аналізу в якості інструменту розробки було обрано нативні інструменти, бо основними вимогами до додатку були: швидкодія, побудова за допомогою стандартних компонентів інтерфейсу платформи та ефективне використання ресурсів системи, беручи до уваги не завжди високу потужність пристроїв кінцевих користувачів.

## Висновки до розділу 2

Розділ надає порівняльну характеристику про існуючі аналоги створюваного програмного продукту. Розглянуто їх недоліки, основними з яких є: орієнтованість тільки на веб, не містять функціоналу, який би дозволяв організовувати процес ведення дипломних робіт, частота використання низька, що спричиняє проблеми при організації навчального процесу.

Також розглянуто нативний та кросплатформний підходи до побудови мобільних додатків, після аналізу переваг та недоліків було обрано нативні засоби розробки.

# 3. ЗАСОБИ ТА ТЕХНОЛОГІЇ РОЗРОБКИ

Після визначення функціоналу додатку наступним шагом є проектування макетів.

## 3.1. Проектування макетів мобільного додатку

У двох словах, каркас або макет для мобільних додатків - це спрощена візуальна концепція майбутнього додатка. Це не дає ніякого уявлення про дизайн, але допомагає зрозуміти, як буде працювати додаток.

Макети відображають:

— розподіл просторів екрану;

— приорітезацію контенту;

— функціонал додатку;

— можливі дії;

— відношення між екранами.

Макети не відображають:

— елементи дизайну;

— кольори;

— фактичні зображення;

— шрифти;

— логотипи.

Якісні макети [10] універсальні для будь-якого типу платформи.

Виділимо основні переваги розробки та використання макетів додатка:

— надають чітке уявлення про те, як буде виглядати додаток та як він буде працювати;

— дозволяють швидко змінювати структуру проекту, не треба перебудовувати код проекту;

— зменшують загальну вартість проекту.

Зібравши продуманий макет можна оцінити й виправити недоліки на ранньому етапі проектування, що буде коштувати значно дешевше, ніж зміни в кінцевому продукті;

— надають розуміння обсягів запланованих робіт.

Макет можна розділити за логічними частинами й оцінити складність кожної окремо;

— макет може виступати в ролі прототипу майбутнього продукту, який можна використати у презентаціях для кінцевого користувача або замовника.

Для побудови макетів найбільш зручними інструментами є: SketchApp, Adobe XD[11].

Серед переваг SketchApp є інтуїтивно зрозумілий інтерфейс, розширений функціонал групування об’єктів, можливість створювати шаблонні елементи. Програма також надає можливість створення інтерактивних переходів між екранами, як зображено на рисунку 3.1.

Має два основних недоліки: працює тільки на Mac OS та має платну ліцензію.

Adobe XD було розроблено як аналог SketchApp. Кросплатформна та безкоштовна, але має обмежені функції проектування в порівнянні з SketchApp.

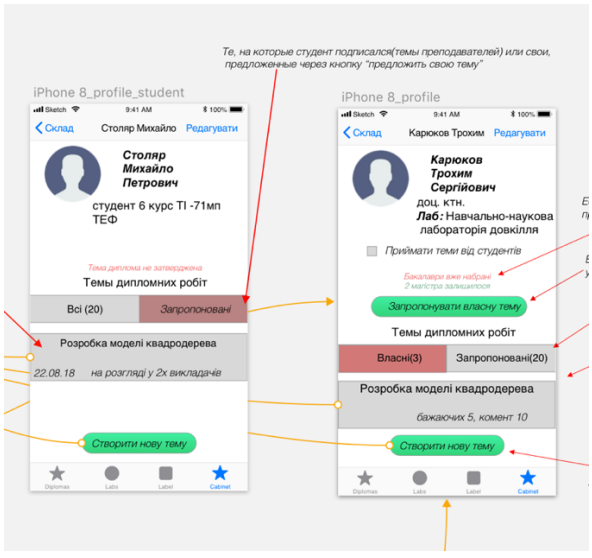



Рисунок 3.1 — Приклад побудови макету

## 3.2. Архітектура системи

Система складається з двох модулів: клієнту на ОС iOS та серверу. Сервер написаний на мові програмування Kotlin[12], що виконується на віртуальній машині Java, клієнтський додаток в свою чергу - на мові Swift.

На стороні сервера використано архітектуру MVC[13] на базі фреймворку Spring Boot та ORM Hibernate для забезпечення зручного об'єктного відображення бази даних та доступу до них.

На стороні клієнта було використано архітектуру MVVMR та RxSwift[14] для зручної роботи із асинхронними потоками даних. Схема загальної архітектури системи зображена на рисунку 3.2.

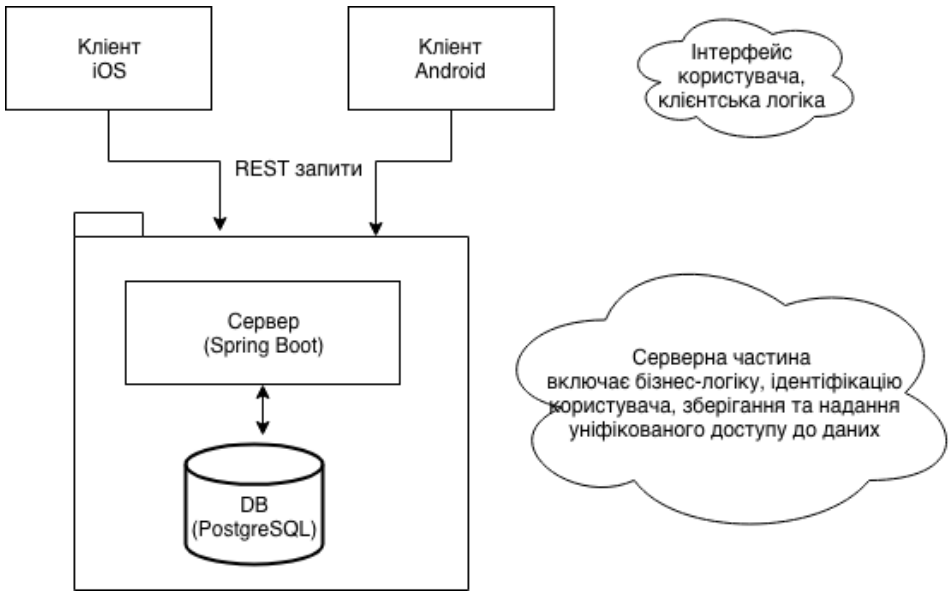



Рисунок 3.2 — Схема клієнт серверної архітектури програмного комплексу

Головним центром програмного модуля є сервер. У ньому зосереджена основна бізнес-логіка. На сервері відбувається ідентифікація користувача для управління його доступом до даних на правами. Сервер є посередником між клієнтами та базою даних, для унеможливлення пошкодження даних.

Згідно з концепціями користувальницької взаємодії(Apple HIG [15]) було вирішено запитувати авторизацію за необхідністю — коли дії користувача стосуються зміни даних, наприклад, створення дипломної роботи. Але логіка доступу

реалізується на рівні серверу, тому що на рівні користувача можлива підміна прав доступу та інші методи неконтрольованого доступу до даних.

Під час користування системою, користувач взаємодіє з клієнтським додатком, iOS в даному випадку. На рівні користувача реалізований інтерфейс, за допомогою якого користувач отримує інформацію про лабораторії та викладачів кафедри, створює та управляє темами дипломних робіт.

Також на користувацькому рівні відбувається попередня обробка даних перед відправленням на сервер і також опрацювання результатів від сервера. На цьому рівні відбувається перший етап автентифікації користувача для обмеження неконтрольованого доступу до програми.

Рівень бази даних зберігає дані та забезпечує консистентність даних за допомогою зовнішніх зв'язків та ключів між даними [16].

## 3.3. Опис архітектури сервера

При побудові сервера першим кроком необхідно визначитися з його архітектурою, кількістю її шарів та відповідальністю кожного.

Сформуємо набір функціоналу, що повинен виконувати сервер:

— обробляти запити клієнта та вертати відповідні відповіді;

— мати механізм обробки помилок, що вертає зрозумілі повідомлення помилок клієнту;

— необхідний засіб управління транзакціями;

— треба обробляти авторизацію та автентифікацію користувача;

— містити бізнес логіку;

— функціонал підключення до зовнішніх ресурсів.

Описаний функціонал можна покрити трьома шарами (рисунок 3.3):

**Web – шар**, який є найвищим шаром веб-додатка. Він відповідає за обробку вхідних запитів користувача та повернення правильної відповіді клієнту.

Web шар також повинен обробляти виключення, створені іншими шарами, бути відповідальним за автентифікацію та виступати в ролі першої лінії захисту від несанкціонованого доступу.

**Шар сервісів** знаходиться нижче web шару та містить сервіси з бізнес логікою.

**Шар репозиторіїв** – найнижчий шар додатку. Необхідний для зв'язку зі сховищем даних.

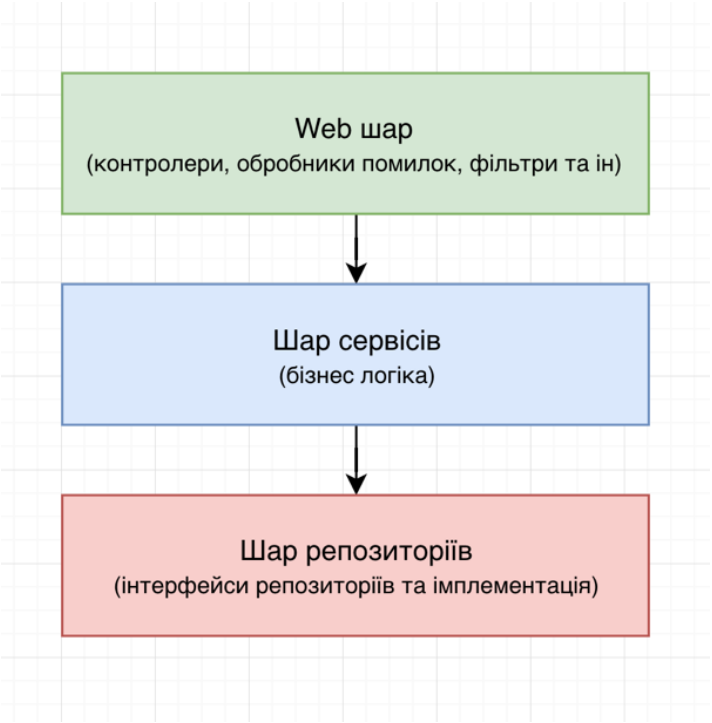



Рисунок 3.3 — Представлення шарів додатку

Компоненти, що належать одному шару можуть використовувати компоненти, що знаходяться **на тому ж рівні**, або **рівнем нижче**.

Наступним кроком буде проектування інтерфейсу кожного шару. Треба визначитися з деякими поняттями [17].

**DTO** – data transfer object – обєкт, який представляє собою звичайний контейнер для даних, ці об'єкти необхідні для передачі даних між різними процесами та шарами додатку.

**Domain model** – складається з об'єктів:

**Entity (сутність)** – обєкт, який представляє з собою якусь сутнісь, та не містить логіки.

Value object – описує властивість сутності, цей обєкт не має власного життєвого циклу чи представлення. Життєвий цикл Value object привязаний до життєвого циклу сутності.

Спроектуємо кожний шар, використовуючи наведені поняття:

— web – шар повинен обробляти тільки DTO;

— шар сервісів приймає DTO обєкти в якості параметрів методів. Він також може працювати. З обєктами domain model, але повертати тільки DTO до web шару;

— шар репозиторіїв працює з сутностями.

Може виникнути питання: “З якою метою в додаток введені DTO, чому не можна просто вертати сутності з шару web?”

— Domain model містить внутрішню модель додатку, якщо відкрити цю модель, то клієнти отримають надлишкові дані в неоптимальній формі, DTO обгортки приховують надлишкові та приватні дані та надають більш зрозуміле API.

— У разі зміни domain model не обовязково змінювати DTO, тому кінцевий клієнт отримає данні тієї ж форми, що і до зміни.

Роздивимось шаблони, що використовуються в Spring.

Шаблон проектування — це архітектура, рішення яке описує яким чином вирішуються задачі, які часто зустрічаються при розробці програмних систем чи додатків[18].

Фреймворк Spring реалізує в собі декілька шаблонів.

Впровадження залежностей / або IoC (Інверсія управління) — це основний принцип процесу розчеплення Spring.

Фабрика — Spring використовує цей шаблон для створення об’єктів beans через посилання на контекст програми.

Патерн Singleton за замовчуванням, це конфігураційний файл (XML), який створюється тільки один раз. Незалежно від того, скільки викликів було зроблено методу getBean(), він завжди буде мати тільки один екземпляр. Це тому, що за замовчуванням всі bean в Spring сінглтони.

Патерн Model View Controller — перевага Spring MVC (рисунок 3.4).

Основна мета застосування цієї концепції полягає в відділенні бізнес-логіки (моделі) від її візуалізації (представлення). За рахунок такого поділу підвищується можливість повторного використання та гнучкість системи.

Частина Model — представлення даних, самі дані, містить знання про предметну область, дані та правила до цих даних, але нічого не знає про контролери та представлення. Модель надає контролеру дані які запрошує користувач, або інша система. У моделі відбувається бізнес-логіка роботи застосунку.

Частина View — представлення, вид, відображення, надає можливість по-різному відображати дані отримані будь-яким способом від моделі, може містити в собі логіку.

Представлення — це кінцевий інтерфейс, з яким взаємодіє користувач. Користувач може передавати дані через представлення.

Частина Controller — управління, зв’язок між моделлю та відображенням.

Основна мета застосування цієї концепції полягає в відділенні бізнес-логіки (моделі) від її візуалізації (представлення). За рахунок такого собі логіку. За рахунок такого поділу підвищується можливість повторного використання та гнучкість системи. Представлення — це кінцевий інтерфейс, з яким взаємодіє користувач. Користувач може передавати дані через представлення.

Частина Controller — управління, зв’язок між моделлю та відображенням.

Основна мета застосування цієї концепції полягає в відділенні бізнес-логіки (моделі) від її візуалізації (представлення). За рахунок такого поділу підвищується можливість повторного використання та гнучкість системи.

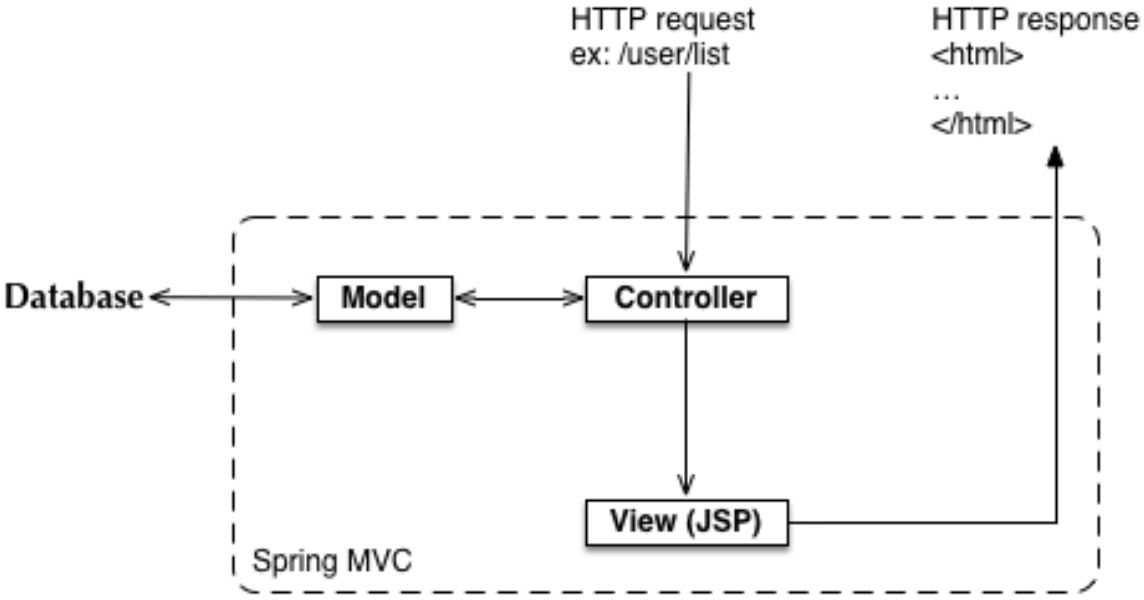



Рисунок 3.4 – Схема роботи MVC шаблону

MVC — один з найпопулярніших шаблонів.

## 3.4. Опис архітектури клієнта

Для реалізації клієнтського додатку на платформі iOS Apple рекомендує використовувати MVC, розглянутий вище.

Але найчастіше в контексті iOS літера M трансформується з Model в Massive, бо View та Controller суміщені в одному класі ViewController, замість двох та бізнес логіка для екрана переноситься або во ViewController що є посуті View в MVC, або перевантажує модель.

Окрім MVC, враховуючи його наведені проблеми, широко використовуються архітектурні патерни MVVM та VIPER (clean architecture [19]). На VIPER базують великі проекти із залежностями, MVVM є одним з найкращіх архітектур для середніх проектів.

MVVM полегшує відокремлення розробки графічного інтерфейсу від розробки бізнес логіки (бек-енд логіки), відомої як модель (можна також сказати, що це відокремлення представлення від моделі). Модель представлення є частиною, яка

відповідає за перетворення даних для їх подальшої підтримки і використання. З цієї точки зору, модель представлення більше схожа на модель, ніж на представлення і оброблює більшість, якщо не всю, логіку відображення даних.

Модель представлення може також реалізовувати патерн медіатор, організовуючи доступ до бек-енд логіки навколо множини правил використання, які підтримуються представленням.

MVVM зручно використовувати замість класичного MVC та йому подібних у тих випадках, коли на платформі, де ведеться розробка, присутнє "зв'язування даних".


Рисунок 3.5 — Схема MVVM шаблону

Шаблон MVVM ділиться на три частини:

— Модель (Model), як і в класичному шаблоні MVC, Модель являє собою фундаментальні дані, що необхідні для роботи застосунку.

— Представлення (View) як і в класичному шаблоні MVC, Вигляд — це графічний інтерфейс, тобто вікно, кнопки тощо.

— Модель вигляду (ViewModel, що означає "Model of View") з одного боку є абстракцією Вигляду, а з іншого надає обгортку даних Моделі, які мають зв'язуватись.

MVVM має два основних недоліки:

— Логіка переходів міститься во View (ViewController), що порушує принцип Single responsibility SOLID та погіршує тестованість.

— ViewModel за своїм призначенням повинен містити бізнес логіку, але має логіку створення модулів (View-ViewModel)

Для позбавлення від цих недоліків необхідно виділити додаткові шари

програми, які б відповідали за переходи та створення модулів.

Було вирішено розробити додатковий клас Router, який містить методи з логікою переходів між модулями та статичний метод для створення поточного модуля. Зберігати посилання на нього з ViewModel як на рисунку 3.6.

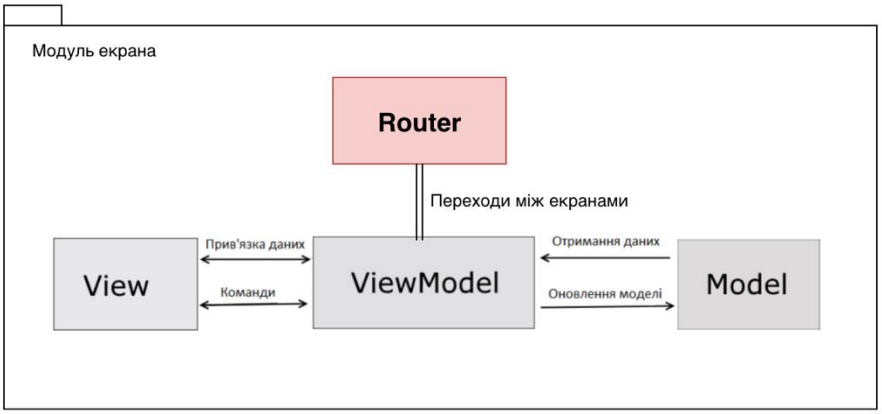



Рисунок 3.6 – Схема MVVMR шаблону

## 3.5. Опис інструментів розробки

**Мова Kotlin** — мова програмування, що працює поверх JMV і розробляється компанією . Позиціонується розробниками як об'єктно-орієнтована мова промислового рівня, при цьому мова повністю сумісна з Java. Зростаюча популярність мови доводиться майже повним заміщенням Java для написання нових Android додатків та підтримки старих, бо є можливість впроваджувати нові функції на Kotlin без переписування програми цілком.

Kotlin працює добре також з Spring Boot, RESTful сервіси можуть бути написані за тією схемою що і на Java. Проте існують деякі незначні відмінності, коли мова йде про визначення градієнтної конфігурації та структури макета проекту, а також про код ініціалізації.

Згідно з блогом Jetbrains, Kotlin використовується у Amazon Web Services, Pinterest, Coursera, має кодову базу, що на більше ніж 90% складаєтсья з Kotlin.

**Spring Framework** — це програмний каркас[20] (фреймворк) з відкритим кодом та контейнери з підтримкою інверсії управління для платформи Java.

Spring Framework не нав'язує якоїсь конкретної моделі програмування, Spring Framework став популярним в спільноті Java як альтернатива, або навіть доповнення моделі Enterprise JavaBean.

Spring Framework складається з кількох модулів, які надають широкий спектр можливостей:

Контейнер Інверсії управління: Конфігурація компонентів додатків і управління життєвим циклом об'єктів Java, здійснюється головним чином через Інверсію управління

Аспектно-орієнтоване програмування: дозволяє реалізувати наскрізні процедури

Доступ до даних: робота з реляційною системою управління базами даних на платформі Java з використанням JDBC і об'єктно-реляційні відображення та інструментів з NoSQL баз даних.

Управління транзакціями: об'єднує кілька API, управління транзакціями та координує операції для Java-об'єктів

Модель-Вигляд-Управління (Model-View-Controller): програмний каркас на основі HTTP сервлета, що забезпечує створення веб-додатків і веб-служб RESTful.

Автентифікація і авторизація: налаштовувані процеси безпеки, які підтримують цілий ряд стандартів, протоколів, інструментів і практик за допомогою підпроекту Spring Security (колишня система безпеки AcerI для Spring).

Віддалене керування: конфігураційний вплив і управління Java-об'єктами для місцевої (локальної) або віддаленої конфігурації через JMX.

Тестування: підтримка класів для написання юніт-тестів та інтеграційних тестів

**Бібліотека Hibernate** — засіб відображення між об’єктами та реляційними структурами (object-relational mapping, ORM) для платформи Java. Hibernate [21] є вільним програмним забезпеченням. Hibernate надає легкий для використання каркас (фреймворк) для відображення між об’єктно-орієнтованою моделлю даних і традиційною реляційною базою даних.

Метою Hibernate є звільнення розробника від значних типових завдань із програмування взаємодії з базою даних. Розробник може використовувати Hibernate як при розробці з нуля, так і для вже існуючої бази даних.

Бібліотека Hibernate піклується про зв'язок класів з таблицями бази даних (і типів даних мови програмування із типами даних SQL [22]), і надає засоби автоматичної побудови SQL запитів й зчитування/запису даних, і може значно зменшити час розробки, який зазвичай витрачається на ручне написання типового SQL і JDBC коду. Hibernate генерує SQL виклики і звільняє розробника від ручної обробки результуючого набору даних, конвертації об'єктів і забезпечення сумісності із різними базами даних.

Забезпечуються можливості з організації відношення між класами «один-до-багатьох» і «багато-до-багатьох». На додаток до управління зв'язками між об'єктами, Hibernate також може керувати рефлексивними асоціаціями, де об'єкт має зв'язок «один-до-багатьох» з іншими примірниками свого власного типу даних.

Hibernate підтримує відображення користувацьких типів значень. Це робить можливим такі сценарії:

— Перевизначення типу за замовчуванням SQL, який Hibernate вибирає при відображенні стовпчика властивості;

— mapping перераховуваного типу Java до колонок БД, так ніби вони є звичайними властивостями;

— mapping однієї властивості в декілька колонок.

**RxSwift** реалізує концепти реактивного та функціонального програмування, які були розроблені спеціалістами з Microsoft та згодом портовані на більшість платформ. RxSwift та RxCocoa - відкриті бібліотеки.

З одного боку, функціональне програмування - це процес побудови програмного забезпечення шляхом складання чистих функцій, уникнення спільного стану, змінних даних та побічних ефектів.

З іншого боку, реактивне програмування - це асинхронна парадигма програмування, що стосується потоків даних та поширення змін.

Спільне функціональне реактивне програмування являє собою комбінацію функціональних та реактивних методів, які можуть представляти елегантний підхід до керування подіями - із значеннями, які змінюються з часом і де споживач реагує на дані, як це відбувається.

Ця технологія об'єднує різні реалізації її основних принципів, деякі автори придумали документ, який визначає загальний словник для опису нового типу програм — маніфест.

**Реактивний маніфест** — це онлайновий документ, який встановлює високі стандарти для програм у галузі розробки програмного забезпечення [23]. Простіше кажучи, реактивними системами є:

Реакційні - системи повинні відповісти своєчасно

Контрольовані повідомленнями — системам слід використовувати асинхронні повідомлення між компонентами для забезпечення взаємодії

Еластичні — системи повинні залишатися чутливими під високим навантаженням або коли деякі компоненти не працюють

Ці концепти реалізовано у бібліотеці за допомогою сутностей двох типів — потоку даних та підписника на потік.

Класи потоків містять великий набір методив для перетворення даних у потоці або для перетворення одних потоків на інші (map, flatmap, filter та ін.)

Класи підписників вміють реагувати на порції нових даних у потоці, на виникнення помилки чи завершення роботи потоку.

Дана бібліотека допомогає виконувати аснихронні операції, особливо коли ці операції необхідно виконувати за певними правилами (послідовно чи паралельно, або за певної умови).

## Висновки до розділу 3

При розробці програмного продукту важливим чинником є правильний вибір засобів програмної реалізації, що впливає на час розробки, якість, надійність та швидкість кінцевого продукту.

В розділі було обгрунтовано необхідність проектування макета інтерфейса майбутнього додатка. Розглянуто загальну архітектуру системи, зв'язок клієнта з сервером.

Проведено аналіз необхідних задач, що має виконувати сервер, у відповідності до цього розроблено трьохшарову архітектуру: web, service, repository.

Розглянуто існуючі архітектури побудови клієнта, обрано MVVM, як найбільш зручну для цього типу додатків та вдосконалено її до архітектури MVVMR.

Також зроблено огляд використовуваних технологій, зокрема RxSwift, мова Kotlin та ін.

# 4. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

Даний сервіс складається з двох модулів — сервер для зберігання та обробки даних користувачів і мобільний додаток, що дозволяє оперувати даними, що знаходяться на сервері (створювати, редагувати та видаляти дані, проводити пошук та відтворення через користувацький інтерфейс).

## 4.1. Опис моделі даних системи

Вхідними даними є дії користувача, що направлені на взаємодію із сутностями (дипломна робота, лабораторія, студент та ін.). Дані зберігаються на віддаленому сервері у реляційній базі даних[24], а задання параметрів — завдання клієнтського додатку.

Вихідна інформація — сутності, що було створено на віддаленому сервері, запит і відображення яких відбуваються у клієнтському додатку.

Одним з перших етапів створення програмного продукту є виділення сутностей а також встановлення зв’язків між ними [25].

До основних сутностей відносяться:

— студент;

— викладач;

— лабораторія;

— напрям лабораторії;

— дипломна робота.

Для визначення зв’язків необхідно виділити обов’язки сутностей.

Було виділено наступні властивості:

— студент і викладач можуть бути авторами теми для дипломної роботи;

— і студент, і викладач мають можливість реєструватися в системі та

редагувати дані про себе;

— дипломна тема має кілька станів. Її можна затвердити чи відхилити, а також запропонувати кільком студентам чи викладачам тощо;

— студенти можуть пропонувати свої теми одразу кільком викладачам, але тільки тим, у яких ще є місце на наукову роботу.

Як результат, було спроектовано схему бази даних, що зображено на рисунку 4.1:

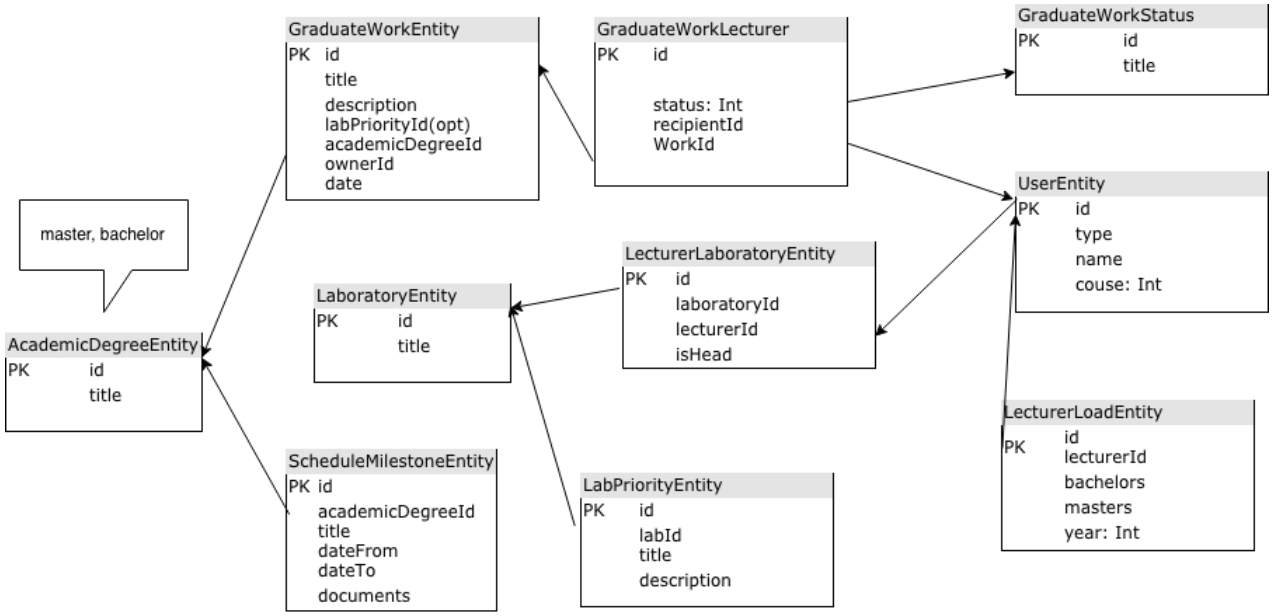



Рисунок 4.1 — Схема бази даних системи

## 4.2. Реалізація структури серверної частини

На основі розглянутої вище трьохшарової архітектури серверу було створено наступну структуру, зображену на рисунку 4.2.

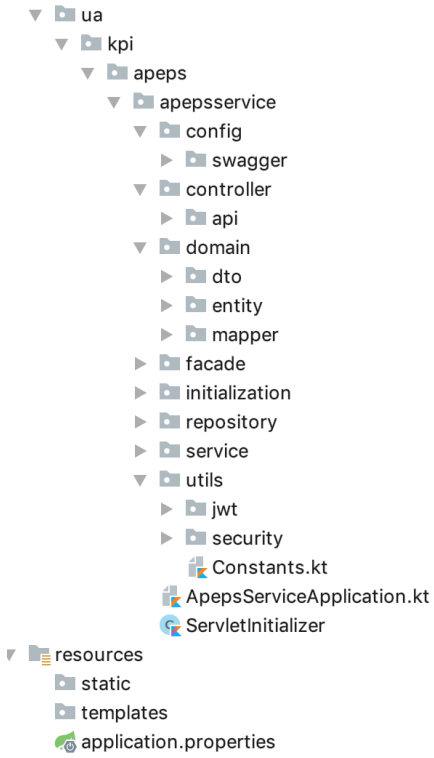



Рисунок 4.2 — Структура пакетів сервера

Основними пакетами, що забезпечують функціонал системи є **controller, service та repository**. [26]

Пакет controller представляє найвищій(web) шаром та містить класи:

— AuthController – має методи реєстрації студента та викладача;

— UsersController – необхідний для роботи з користувачами системи, а саме: отримувати детальну інформацію про користувача за певним id, оновляти інформацію користувача. Використовується клієнтом для отримання інформації у разі переходу на екран профіля;

— LaboratoryController — може вертати всі лабораторії, за певним id, також вертає викладачів за лабораторією та інформацію по напрямкам лабораторії за певним id.

— GraduateWorksController — контроллер роботи з дипломами, дозволяє створювати теми дипломних робіт, оновлювати їх, отримувати створені роботи за id автора, ті роботи, що запропоновані викладачеві за id, методи, що дозволяють попонувати та відхіляти роботи

— ScheduleController — містить методи, що вертають розклад здачі робіт.

—

Роздивимось наступний шар — шар сервісів.

Сервісів представлено чотири:

— GraduateWorksService;

— LabService;

— UserService;

— ScheduleService.

Вони реалізують бізнес логіку, що потребують контролери.

Треба зауважити, що наведені класи — інтерфейси, до кожного сервісу відповідна йому інплементація. Spring широко використовує dependency injection [27], завдяки чому виконується останній принцип SOLID — інверсії залежностей, тобто під інтерфейс може бути підставлена будь-яка реалізація, в тому числі і для тестування.

За обовязки обробки авторизації та автентифікації відпофідають авторизаційні фільтри та сервіс UserDetailsService [28].

**JWTAuthenticationFilter** необхідний для перевірки даних авторизації та формування Json Web Token(JWT) у разі валідної авторизації.

В токен записується юзернейм користувача та дата закінчення дії токена.

**JWTAuthorizationFilter** кожний запит, що потребує авторизації, перевіряє на наявність валідного JWT. Токен є валідним, якщо його секретне слово відповідає секретному слову сервера та дата закінчення дії більша за поточну.

Інтерфейс UserDetailsService використовується для отримання даних про користувача. В ньому є метод loadUserByUsername() який шукає користувача за юзернеймом.

В цей сервіс за допомогою dependency injection постачається репозиторій UserRepository для пошуку користувача в БД.

Відповідальним за налаштування авторизації є клас WebSecurity,
Який налаштовує шляхи, за якими потрібна чи не потрібна авторизація.
Клієнт взаємодіє з сервером через API, яке має бути задокументованим.

Є два способи документації API: вручну або автоматично.

Автоматичний спосіб має головну перевагу: оновлюється автоматично при зміні API.

Було вирішено скористатися бібліотекою Swagger 2 [29], що автоматизує опис API, результат зображено на рисунку 4.3.

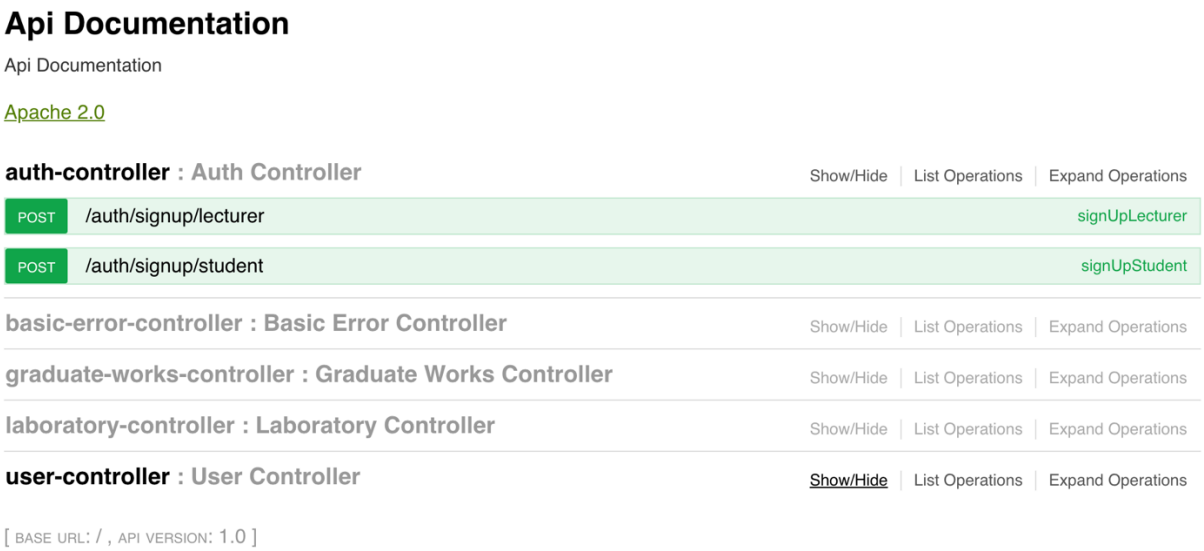



Рисунок 4.3 — Згенерована документація

В системі існує два типи користувачів: викладач та студент, які відрізняються полями, виконують різні функції, але мають спільну базу – є типом людина.

Реляційні бази даних не мають прямого способу відображення ієрархій класів на таблиці бази даних.

Для вирішення цього питання, специфікація JPA [30] надає кілька стратегій:

— **одиночна таблиця** — об'єкти з різних класів із загальним предком поміщаються в єдину таблицю;

— **приєднана таблиця** — у кожного класу є своя таблиця, і запит підкласу об'єкта вимагає об'єднання таблиць;

— **таблиця на клас** — всі властивості класу знаходяться в його таблиці, тому приєднання не потрібно.

Стратегія **одиночна таблиця** створює одну таблицю для всіх надкласів. Ця стратегія йде за замовчуванням.

Визначити стратегію можна додаванням анотації @Inheritance з типом до суперкласу.

Оскільки записи для всіх нащадків будуть знаходитися в тій же таблиці, Hibernate потребує спосіб диференціювати їх [31].

За промовчанням це виконується через стовпець дискримінатора під назвою DTYPE, що має назву класу в якості значення.

Для налаштування стовпця дискримінатора, можна використовувати анотацію @DiscriminatorColumn. Використаємо “student” та “lecturer”.

Недолік цього підходу в зберіганні різних полів всіх нащадків в одній таблиці, тобто при створенні запису одного типу поля, що не належать до цього типу будуть пусті та займатимуть пам'ять.

**Приєднані таблиці —** використовуючи цю стратегію, різні поля надкласів виносяться в окремі таблиці. Звязуються з базовою за допомогою ключів.

Недоліком є те, що при вибірці даних виконується обєднання таблиць, зо призводить до погіршення швидкодії.

Стратегія **таблиця на клас** відображає кожну сутність в окремій таблиці, базові поля дублюються в кінцевих таблицях.

Для отримання даних використовується операція UNION, що призводить до погіршення швидкодії.

Було обрано стратегію **одиночна таблиця** як найбільш швидку.

## 4.3. Опис підходу до архітектури клієнта

“Код пишеться для людей, а не для машин, бо машини його читають, а люди розуміють”.

Клієнт був розроблений, враховуючи принципи SOLID, шари додатку взаємодіють через протоколи(інтерфейси), що дозволяє незалежно окремо тестувати кожен з шарів. Система поділена на функціональні модулі, які будуються через власні роутери [32].

Модулі можуть бути скомбіновані довільним чином в більш великі структури, бо дотримано принцип dependency inversion. З накопиченням класів новий

функціонал можна буде набирати з вже існуючого, змінюючи логіку роботи між частинами, що прискорить розробку та покращить перевикористовування коду.

Щодо організації коду в проекті, існує багато варіацій на цю тему. В поточному проекті всі класи, що відносяться до незалежної функціональної частини були розміщені в своєму модулі для простоти розуміння і інтуїтивності навігації.

IOS додаток структурно розділено на такі частини:

— Modules - модулі, що відображують конкретні екрани з бізнес логікою (рисунок 4.4)

— Network - частина, що містить шар роботи з API сервера

— Entities - сутності в системі(Дипломна робота, Користувач та ін.)

— Resourses - xml розмітка та медіафайли

— Utils - допоміжні класи, такі як валідатори, фабріки та ін.

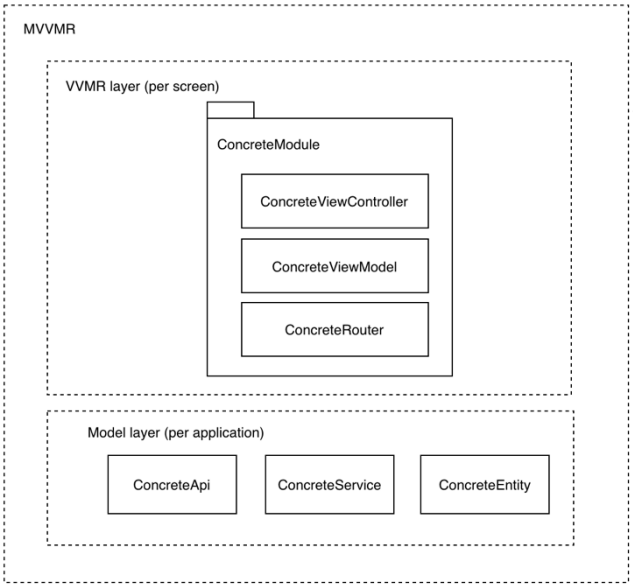



Рисунок 4.4 — Схема організації класів

Роздивимось частину Network. RestApiImpl, що закритий за протоколом RestApi, проксує виклики до RESTClient. RESTClient використовує сторонній фреймворк роботи з мережею “Alamofire”, RestApi має реактивні обгортки методів, що дозволяє підставити в реалізації методів будь-який постачальник даних.

RestClient має методи, які приймають обєкти типу Endpoint, що містять інформацію про запит, рисунок 4.5.

```
init(method: Method = .get,
     path: Path,
     parameters: Parameters? = nil,
     headers: Headers? = nil,
     encoding: ParameterEncoding = JSONEncoding.default,
     decode: @escaping (Data) throws -> Response) {
    self.method = method
    self.path = path
    self.parameters = parameters
    self.headers = headers
    self.encoding = encoding
    self.decode = decode
}
```

Рисунок 4.5 — Конструктор класа Endpoint

Набір Endpoint міститься в класі API у вигляді статичних методів.

Для RestAPI є також тестова імплементація RestAPITest, для тестування нереалізованого на той момент функціоналу сервера [33].

В частині Entities знаходяться сутності, якими оперує система.

Особливою сутністю є UserEntity, яка реалізує два протокола LecturerEntity та StudentEntity. Містить всі поля студента та викладача. На першому етапі проектування системи було розглянута ідея створити базовий клас UserEntity та два нащадки: StudentEntity та LecturerEntity, але враховуючи те, що є методи сервера, які вертають користувача за його id, то наперед невідомо його тип. В разі необхідності можно превести клас до протокола відповідного до типу користувача, наприклад , при переході на детальний екран користувача.

Для списків користувачів розроблено протокол [34] UserRepresentable, який, в свою чергу, імплементується UserEntity, LecturerEntity та StudentEntity.

Resources – директорія, що містить набір зображень інтерфейсу та storyboards – xml файли, за допомогою яких будується інтерфейс системи (рисунок 4.6)

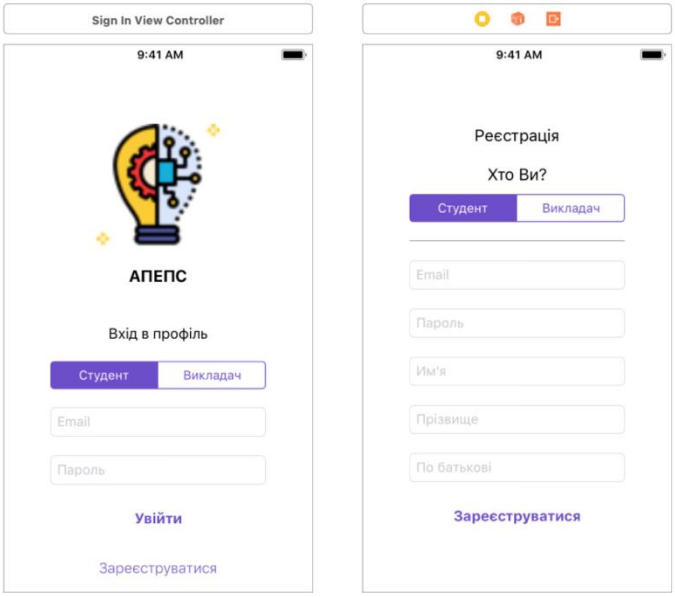



Рисунок 4.6 — Storyboard авторизації

Utils містить допоміжні класи:

— StoryboardFactory — дозволяє зручно створювати ViewController;

— Localization — допоміжний клас, який дозволяє змінити мову інтерфейсу без перезапуску додатку [35].

За правилами iOS, мова додатку повинна співпадати з мовою девайса, але в деяких випадках користувачеві зручно використовувати іншу мову додатку.

В класі Localization є статичне поле localeIdentifier, задаючи яке, можна змінити локалізацію строк, що надає клас NSLocalizedString. Власний метод NSLocalizedString класа Localization. Зчитує файл з заданою в localeIdentifier локалізацією та вертає відповідні переклади.

Треба мати на увазі, що після зміни мови необхідно перезавантажити інтерфейси в стеці екрана.

Каталог Module зберігає всі функціональні екрани або модулі, кожний з яких складається з ViewController, ViewModel, Router. (рисунок 4.7)

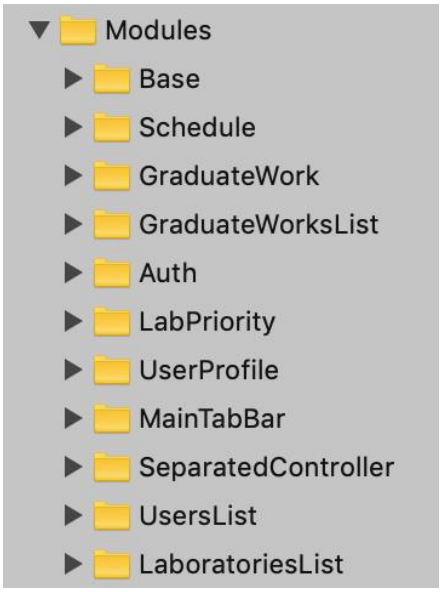



Рисунок 4.7 — Модулі додатку

ViewController та ViewModel повязані засобами реактивного програмування. ViewModel є пасивною, після ініціалізації відповідного ViewController (метод viewDidLoad) визивається метод populate у viewmodel в який передаються дії viewcontroller, в свою чергу, viewcontroller також підписаний на зміни у viewmodel [36].

В розробленій архітектурі MVVMR є основний недолік – за збірку модулів відповідає роутер, як зображено на рисунку 4.8.

```
class SignUpRouter {
    weak var viewController:UIViewController!
    static func assembleModule() -> UIViewController {
        let vc: SignUpViewController! = StoryboardFactory.Auth.signUp.instantiateViewController()
        vc.title = Localization.NSLocalizedString("Реєстрація", comment: "")
        let router = SignUpRouter()
        let vm = SignUpViewModel(api: RestAPIImpl.shared, bgScheduler:
            ConcurrentDispatchQueueScheduler(qos: .userInitiated))
        router.viewController = vc
        vm.router = router
        vm.view = vc
        vc.vm = vm
        return vc
    }
```

Рисунок 4.8 — Метод збірки модуля SignUp

Тобто роутер несе відповідальність не тільки за навігацію [37]. Це не відповідає принципу single responsibility (SOLID). Окрім цього, в кожному роутері необхідно

було надавати конкретні реалізації сервісів, це призвело до дублювання коду і проблем при зміні сервісу у всьому додатку [38].

Було вирішено винести код для створення модулів в окремі фабрики, кожна фабрика б обєднувала деякий логичний набір модулів, наприклад, модулі для авторизації. Для всіх фабрик була створена обєднуюча фабрика (AssemblyFactory).

BaseAssembly – базовий клас для всіх фабрик, містить в собі посилання на обєднуючу фабрику AssemblyFactoryImpl та фабрику сервісів, де сервіси – шар API, валідатори та ін.

Створення модулів та сервісів було розбито по фабрикам [39]. Було також створено базовий клас BaseRouter, який має методи переходів між екранами та посилання на фабрики сервісів та модулів.

В результаті рефакторингу було дотримано принцип single responsibility та задача роутерів обмежилася переходом між екранами, використовуючи зібрані модулі, повернуті фабрикою [40].

## Висновки до розділу 4

Розділ містить інформацію про внутрішній устрій системи, розглянуто концептуальну модель бази даних, методи відображення наслідування в реляційній базі даних. На клієнтському додатку код створення модулів винесено в фабрики.

# 5. МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Розроблена система працює на операційній системі iOS, інсталювати можна через сервіс AppStore після публікації.

## 5.1. Інсталяція та системні вимоги

Для клієнтського додатку:

— система може працювати на смартфоні з iOS 9.0;

— встановити додаток можна з AppStore після публікації.

Для серверної частини:

— IBM-сумісний комп'ютер з процесором класу Intel Core та вище й тактовою частотою 1.5 Гц.

— Об'єм оперативної пам'яті не менше 1Гб.

— Доступ до мережі Інтернет зі швидкістю не менше 1Мбіт/сек.

— 1Гб дискового простору

— Встановлений Docker

— Сервер PostgreSQL

— JVM версії не нижче 1.8

## 5.2. Графічний інтерфейс користувача

Інтерфейс програми складається з декількох "табів". Кожен з них створений для керування відповідною частиною програми.

На рисунку 5.1 зображено екран зі списком лабораторій. Це перший екран, який побачить користувач при запуску додатку.

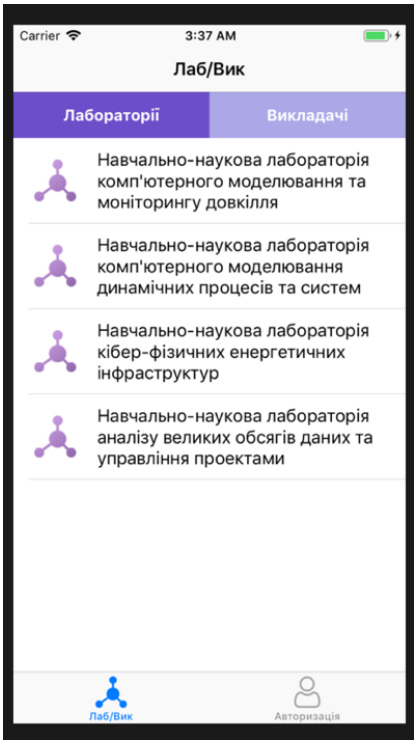



Рисунок 5.1 —Екран з лабораторіями

При переході на таб “профіль”, користувач побачить екран авторизації(рисунок 5.2), якщо не був попередньо авторизований.


Рисунок 5.2 — Екран авторизації

Логін — email користувача. Логін і пароль мають валідацію регулярними виразами.

Якщо користувач введе некоректні значення у поле, воно відобразить помилку.

Після успішного логіну користувач потрапляє до свого профілю.

На рисунку 5.3 зображено екран перегляду профілю користувачів.

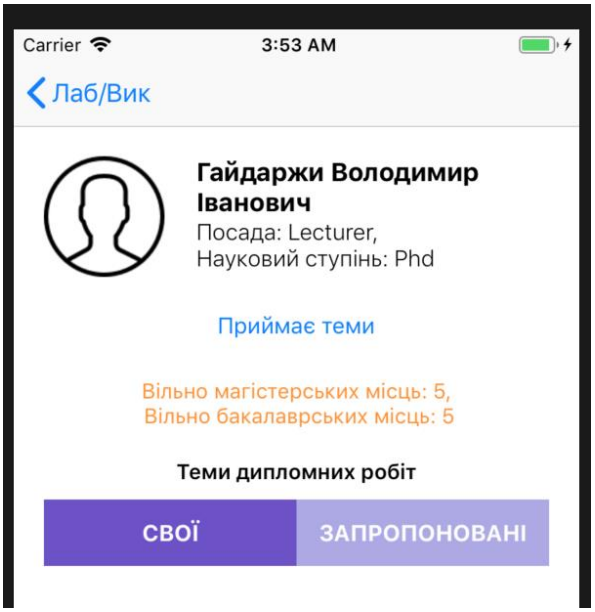



Рисунок 5.3 —Екран профілю

На даному екрані зображена вся інформація про користувача, а саме:

— фотографія;

— ПІБ;

— статус (науковий ступінь для викладачів, група для студентів);

— список дипломних робіт (своїх та запропонованих).

При натисненні на кнопку “Створити роботу” користувач переходить на екран створення нової теми. Екран створення нової теми зображено на рисунку 5.4

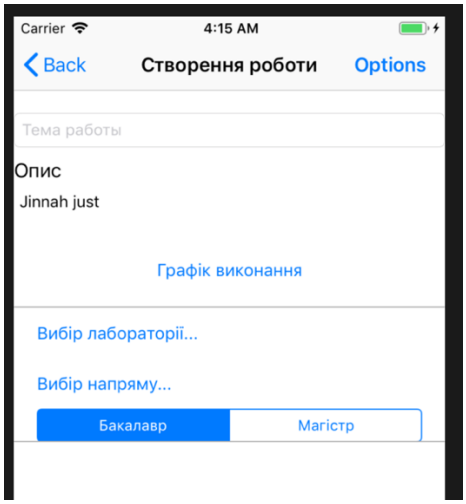



Рисунок 5.4 — Екран створення нової теми

Для створення нової теми необхідно ввести таки дані:

— тема;

— опис теми.

Також необхідно обрати лабораторію та напрям у ній, визначити ступінь роботи (бакалавр, магістр)

Теми можуть бути створені як студентами, так і викладачами, але різниця в тому, що викладачі можуть створювати теми тільки в рамках своїх лабораторій, а студенти — у будь-якій.

Вкладка “лабораторії/викладачі” (рисунок 5.5) містить розділений список з всіма лабораторіями на першій вкладці та всіма викладачами на другій, студент може швидко знайти викладача, до якого хоче потрапити на наукову роботу.

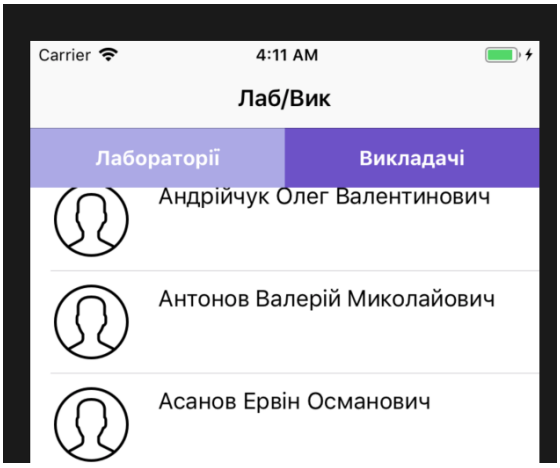



Рисунок 5.5 — “лабораторії/викладачі”

З даного екрану можна перейти на профіль викладача, де можна побачити його теми, теми, що були запропоновані іншими студентами та завантаженість, а також підписатись на його теми. Після підписки на тему викладач може підтвердити вас у якості виконавця. При перегляді дипломної роботи також видно усіх підписників на неї, а також стан їх підписки (розглядаються, прийняті, затверджені чи відхилені).

Сам викладач не має можливості запропонувати свою тему конкретному студенту. Викладач тільки створює їх і чекає на підписки, після чого розглядає їх. Студент, у свою чергу, має можливість самостійно підписувати викладача на свою тему, тобто запропоновувати її.

Важливим моментом є те, що при затвердженні теми з одним викладачем, всі підписки інших тем до інших викладачів зникають, лічильник вільних місць викладача, що затвердив тему, зменшується. Якщо у студента є хоч одна затверджена тема, то він не може запропоновувати теми викладачам.

Для ознайомлення з кафедрою та її лабораторіями необхідно перейти на першу вкладку таба “лабораторії/викладачі” (рисунок 5.6).

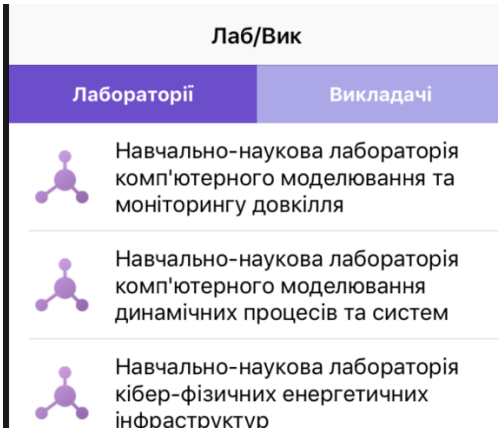



Рисунок 5.6 — “лабораторії/викладачі”

А з неї на вкладку напрямків та викладачів, що належать до даної лабораторії (рисунок 5.7)

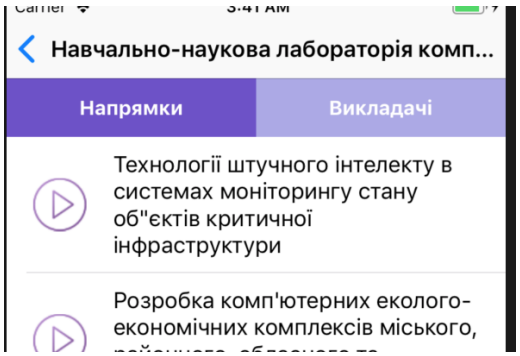



Рисунок 5.7 — вкладка “Напрямки лабораторії”

Користувач має можливість детальніше почитати про напрямок лабораторії на екрані “напрямок”, де є короткий опис напрямку. Вкладка “лабораторії” допомагає студенту визначитись із тим, куди краще віднести його тему.

Для перегляду розкладу студенту необхідно перейти на другий таб програми, за допомогою перемикача можна також обрати рівень. (рисунок 5.8)

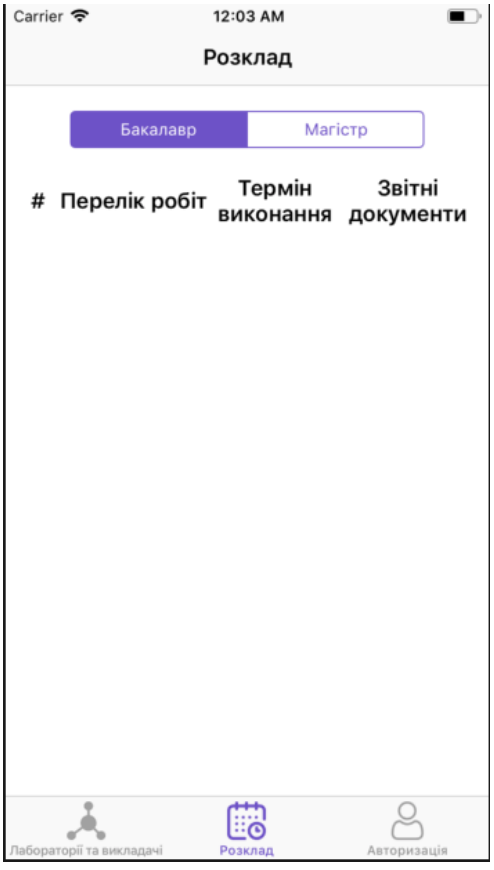



Рисунок 5.8 — Розклад термінів виконання

## Висновки до розділу 5

Розділ містить інформацію про системні вимоги як для клієнтської так ідля серверної частини, а також можливі сценарії взаємодії користувача з інтерфейсом:

— список лабораторій;

— список викладачів;

— профіль;

— розклад;

— список напрямів лабораторії.

# 6. СТАРТАП ПРОЕКТ

Розділ містить проведення маркетингового аналізу стартап-проекту із ціллю визначення можливості його впровадження на ринку та способів реалізації цього впровадження. Проведення аналізу складається з кроків, що описано нижче.

## 6.1. Опис ідеї проекту

Опис ідеї представляється у вигляді наступних таблиць:

— зміст запропонованої ідеї;

— потенційні напрямки застосування;

— основні вигоди для користувача продукту;

— особливості продукту у порівнянні із конкурентами.

Перші три пункти подаються у вигляді таблиці 6.1 і дають цілісне уявлення про зміст ідеї та потенційні напрямки застосування.

Таблиця 6.1. Опис ідеї стартап-проекту

| **Зміст ідеї** | **Напрямки застосування** | **Вигоди для користувача** |
|---|---|---|
| Організація наукової та освітньої діяльності кафедри | 1. Створення електронних черг | 1. Уніфікація певних робочих процесів |
| | 2. Організація дистанційного спілкування між студентом та викладачем | 2. Надання актуальної інформації |
| | 3. Документування та ведення статистики на кафедрі | 3. Документальне підтвердження виконаних завдань |

Аналіз потенційних техніко-економічних переваг ідеї (тобто особливості продукту) порівняно із конкурентними продуктами передбачає:

— визначення переліку техніко-економічних властивостей та характеристик ідеї;

— визначення проектів-конкурентів, що вже існують на ринку;

— збір інформації щодо значень техніко-економічних показників для ідеї даного проекту та конкурентних проектів згідно з визначеним вище переліком;

— проведення порівняльного аналізу показників із наступними значеннями: а) гірші значення (W, слабкі); б) аналогічні (N, нейтральні) значення; в) кращі значення (S, сильні) (таблиця 6.2). [**Ошибка! Источник ссылки не найден.**]

Таблиця 6.2. Визначення сильних, слабких та нейтральних характеристик

| **No п/п** | | **товари/концепції конкурентів** | | | |
|---|---|---|---|---|---|
| | | **Мій проект** | **Campus KPI** | **Online Tables** | **Messenger** |
| 1 | W слабка сторона | Відсутність веб-версії | Відсутність мобільної версії | Відсутність вузької направленості | Відсутність вузької направленості |
| 2 | | Неможливість прикріпляти файли | Жорстка прив’язка до конкретного вузу | Немає ролей у акаунтів | Багато зайвого функціоналу |
| 3 | N нейтральна сторона | Можливість вирішення питань з дипломними роботами | Можливість бродкастити зміни на своїй сторінці | Можливість своєчасно оновлювати дані | Можливість пошуку ввикладачів що зареєстровані |
| 4 | S сильна сторона | Уніфікований простий механізм взаємодії | Велика база користувачів | Просунутий механізм для роботи з таблицями | Нотіфікації |
| 5 | | Орієнтація на дві мобільні платформи | Розширений функціонал із направленням на університетську діяльність | Можливість переведення даних в локальний файл | Можливість розширеного спілкування |

## 6.2. Технологічний аудит ідеї проекту

В межах даного підрозділу проведено технологічний аудит, за допомогою якого можна реалізувати ідею проекту. Для визначення технологічної здійсненності ідеї проекту необхідно проаналізувати наступні складові (таблиця 6.3):

— технологія виготовлення продукту згідно з ідеєю проекту;

— пошук визначених технологій;

— визначення ступеня доступності технологій авторам проекту.

За результатами аналізу таблиці робиться висновок щодо можливості технологічної реалізації проекту: так чи ні, а також технологічного шляху, яким це доцільно зробити (з поміж названих технологій обираються такі, що доступні авторам проекту та є наявними на ринку).

Таблиця 6.3. Технологічна здійсненність ідеї проекту

| **No п/п** | **Ідея проекту** | **Технології її реалізації** | **Наявність технологій** | **Доступність технологій** |
|---|---|---|---|---|
| 1 | Нативний інтерфейс користувача | Android Studio, XCode | Наявна | Доступна безкоштовно |
| 2 | Серверна частина | Spring+Hibernate | Наявна | Доступна безкоштовно |
| 3 | Реляційна база даних | PostgreSQL | Наявна | Доступна безкоштовно |
| 4 | Реалізація чистої архітектури | Dagger, RxJava, Kotlin, Retrofit, RxSwift, Swift | Наявна | Доступна за вільною ліцензією Apache, MIT |
| 5 | Сервіси для нотіфікацій | Firebase | Наявна | Доступна за гроші згідно з тарифними планами |
| Висновок: проект реалізувати можливо.<br>Обрана технологія реалізації ідеї проекту: Нативний інтерфейс користувача, реалізація серверної частини та чистої архітектури, без нотіфікацій. | | | | |

## 6.3. Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, необхідних та наявних під час впровадження проекту на ринку, а також ринкових загроз, які можуть цьому перешкодити, необхідне для того щоб спланувати напрями розвитку проекту із урахуванням стану ринку (включаючи потреби потенційних клієнтів та пропозиції конкурентних проектів).

Для цього в першу чергу необхідно провести аналіз попиту: наявність, обсяг, та динаміка розвитку ринку (таблиця 6.4).

Таблиця 6.4. Попередня характеристика потенційного ринку стартап-проекту

| **No п/п** | **Показники стану ринку (найменування)** | **Характеристика** |
|---|---|---|
| 1 | Кількість головних гравців, од | 3 |
| 2 | Загальний обсяг продаж, грн/ум.од | 2000 грн |
| 3 | Динаміка ринку (якісна оцінка) | Зростає |
| 4 | Наявність обмежень для входу (вказати характер обме- жень) | Немає |
| 5 | Специфічні вимоги до стандартизації та сертифікації | Правила Вузів |
| 6 | Середня норма рентабельності в галузі (або по ринку), % | 60 % |

Середня норма рентабельності в галузі (або по ринку) порівнюється із банківським відсотком на вкладення. За умови, що останній є вищим, можливо, має сенс вкласти кошти в інший проект.

За результатами аналізу таблиці робиться висновок щодо того, чи є ринок привабливим для входження за попереднім оцінюванням.

Надалі визначаються потенційні групи клієнтів, їх характеристики, та формується орієнтовний перелік вимог до товару для кожної групи (таблиця 6.5).

Таблиця 6.5. Характеристика потенційних клієнтів стартап-проекту

| **No п/п** | **Потреба, що формує ринок** | **Цільова аудиторія (цільові сегменти ринку)** | **Відмінності у поведінці різних потенційних цільових груп клієнтів** | **Вимоги споживачів до товару** |
|---|---|---|---|---|
| 1 | Налагодження взаємодії між викладачами та студентами | ВУЗИ та інші учбові заклади | особливості купівлі: компанії заключають довготривалі договори, а стартапери віддають перевагу пробному терміну стандарти: в компаніях архітектором баз даних та мобільними розробниками може виступати декілька людей, в стартапах зазвичай це одна людина | стабільність роботи Невисока ціна (корпоративна закупівля ліцензії на цілий відділ) Наявність документації Підтримка мобільних платформ |

Після визначення потенційних груп клієнтів проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (таблиці 6.6-5.7).

Таблиця 6.6. Фактори загроз

| **No п/п** | **Фактор** | **Зміст загрози** | **Можлива реакція компанії** |
|---|---|---|---|
| 1 | Підходить для кафедр з орієнтацією на лабораторії | Для кафедр з іншою структурою необхідні зміни та доповнення | Додавання альтернативного групування (не за лабораторіями) |
| 2 | Власний формат зберігання | При необхідності змінити інструмент, компанії доведеться це робити вручну, оскільки використовується власна структура | Додавання можливості автоматизованого експорту в різні типи сховищ |

| | | | |
|---|---|---|---|
| | | збереження даних | |
| 3 | Обмеженість функцій | Інструмент обмежений наявними функціями і не має деяких функцій, які мають конкуренти (наприклад: індекси) | Додавання нових функцій за потреби |

Таблиця 6.7. Фактори можливостей

| **No п/п** | **Фактор** | **Зміст можливості** | **Можлива реакція компанії** |
|---|---|---|---|
| 1 | Популярність мобільних платформ | Мобільна індустрія наразі друга за розвитком після веб-індустрії в постійно набирає обертів | Вихід на мобільний ринок |
| 2 | Потреба у автоматизації процесів | Ідея з'явилась через існуючі потреби на кафедрі. Вона актуальна скрізь | Надання інструменту для організації наукової роботи кафедри |
| 3 | Відсутність повноцінних альтернатив | Існуючі альтернативи не надають можливості контролювати процес ведення дипломів безпосередньо | Реалізація відсутнього функціоналу |

Надалі проводиться аналіз пропозиції: визначаються загальні риси конкуренції на ринку. Аналіз пропозиції необхідно виконати аналізуючи існуючі види конкуренцій.

Пропозиції повинні відповідати на питання "Як просувати продукт".

Аналіз пропозицій зображено на таблиці 6.8.

Таблиця 6.8. Ступеневий аналіз конкуренції на ринку

| **Особливості конкурентного середовища** | **В чому проявляється дана характеристика** | **Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)** |
|---|---|---|
| 1. Вказати тип конкуренції - монополія/олігополія/ монополістична/чиста | чиста | Прямі договори з стартапами, презентація продукту на виставках |
| 2. За рівнем конкурентної боротьби<br>- локальний/національний/... | національний | Публікація статей на міжнародних сайтах |
| 3. За галузевою ознакою - міжгалузева/ внутрішньогалузева | внутрішньогалузева | Розвивати напрямки, нерозвинуті конкурентами |
| 4. Конкуренція за видами товарів:<br>- товарно-родова<br>- товарно-видова<br>- між бажаннями | товарно-видова | Розповідати про свої переваги перед конкурентом у цій галузі |
| 5. За характером конкурентних переваг<br>- цінова / нецінова | нецінова | Надання функцій, які не надають конкуренти |
| 6. За інтенсивністю<br>- марочна/не марочна | марочна | Надання функцій, які не надають конкуренти |

Після аналізу конкуренції проводиться більш детальний аналіз умов конкуренції в галузі (таблиця 6.9). [**Ошибка! Источник ссылки не найден.**]

Таблиця 6.9. Аналіз конкуренції в галузі за М. Портером

| **Складові аналізу** | **Прямі конкуренти галузі** | **Потенційні конкуренти** | **Постачальники** | **Клієнти** | **Товари-замінники** |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| | Campus | Excel online, Messenger | Мінімізація витрат часу постачальників | Контроль якості | Лояльність споживачів |
| **Висновки:** | Визначити інтенсивність конкурентної боротьби з боку прямих конкурентів | Є можливості виходу на ринок, оскільки існуючі рішення не надають потрібних переваг | Постачальники підлаштовуються під ринок | Клієнти диктують вимоги згідно з умовами експлуатації | Обмеження для роботи на ринку через товари замінники |

| | | | | | |
|---|---|---|---|---|---|
| | | Системе формування сценаріїв | Мінімізація витрат часу постачальників | Контроль якості | Лояльність споживачів |
| Висновки: | Визначити інтенсивність конкурентної боротьби з боку прямих конкурентів | Є можливості виходу на ринок, оскільки існуючі рішення не надають потрібних переваг | Постачальники підлаштовуються під ринок | Клієнти диктують вимоги згідно з умовами експлуатації | Обмеження для роботи на ринку через товари замінники |

На основі аналізу конкуренції, проведеного в п. 3.5 (таблиця 6.9), а також із урахуванням характеристик ідеї проекту (таблиця 6.2), вимог споживачів до товару

(таблиця 6.5) та факторів маркетингового середовища (таблиця 6.6-6.7) визначається та обґрунтовується перелік факторів конкурентоспроможності. Аналіз оформлюється за таблицею 6.10. [**Ошибка! Источник ссылки не найден.**]

Таблиця 6.10. Обґрунтування факторів конкурентоспроможності

| **No п/п** | **Фактор конкурентоспроможності** | **Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)** |
|---|---|---|
| 1 | Підтримка декількох платформ | Існуючі конкуренти орієнтовані на веб |
| 2 | Орієнтація на вузьку галузь | Існуючі конкуренти не орієнтовані на вирішення питань, що є пріоритетними для даного проекту |

За визначеними факторами конкурентоспроможності (таблиця 6.10) проводиться аналіз сильних та слабких сторін стартап-проекту (таблиця 6.11). [**Ошибка! Источник ссылки не найден.**]

Таблиця 6.11. Порівняльний аналіз сильних та слабких сторін

| **No п/п** | **Фактор конкурентоспроможності** | **Бали 1-20** | **Рейтинг товарів-конкурентів у порівнянні з Database Generator (даним продуктом)** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | **-3** | **-2** | **-1** | **0** | **1** | **2** | **3** |
| 1 | Підтримка декількох платформ | 20 | + | | | | | | |
| 2 | Генерація додаткових зручних методів | 10 | | | + | | | | |

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних та слабких сторін, загроз та

можливостей (таблиця 6.12) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (таблиця 6.11). [**Ошибка! Источник ссылки не найден.**]

Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища.

Таблиця 6.12. SWOT-аналіз стартап-проекту

| | |
|---|---|
| **Сильні сторони:**<br>Орієнтація на декілька мобільних платформ | **Слабкі сторони:**<br>Відсутність нотифікацій |
| **Можливості:**<br>Популярність мобільних платформ<br>Потреба в урегулюванні робочих процесів на кафедрі<br>Відсутність повноцінних альтернатив | **Загрози:**<br>Низький рівень кастомізації<br>Обмеженість функцій |

Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками (прогнозованими результатами) впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення. [**Ошибка! Источник ссылки не найден.**]

Таблиця 6.13. Альтернативи ринкового впровадження стартап-проекту

| **No п/п** | **Альтернатива (орієнтовний комплекс заходів) ринкової поведінки** | **Ймовірність отримання ресурсів** | **Строки реалізації** |
|---|---|---|---|
| 1 | Орієнтація поточної моделі на ринок стартаперів | 10 % | 40 год |
| 2 | Орієнтація поточної моделі на ринок державних установ | 70 % | 240 год |
| 3 | Орієнтація поточної моделі на ринок ентерпрайз | 60 % | 160 год |

| | | | |
|---|---|---|---|
| 4 | Переорієнтація на генерацію серверної частини | 80 % | 120 год |
| 5 | Переорієнтація на веб-розробку | 60 % | 100 год |
| 6 | Переорієнтація на перенесення БД з одних платформ на інші | 60 % | 60 год |

Альтернатива, де отримання ресурсів є більш простим та ймовірним – №2 " Орієнтація поточної моделі на ринок державних установ ", що становить 70 відсотків. Це значення перевищує інші альтернативи.

Альтернатива, де строки реалізації є більш стислими – №1 " Орієнтація поточної моделі на ринок стартаперів". Терміни реалізації в цьому разі становлять лише 40 годин.

## 6.4. Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (таблиця 6.14).

За результатами аналізу потенційних груп споживачів автори ідеї обирають цільові групи, для яких вони пропонуватимуть свій товар. Також на основі цих даних визначається стратегія охоплення ринку. [**Ошибка! Источник ссылки не найден.**]

Таблиця 6.14. Вибір цільових груп потенційних споживачів

| **No п/п** | **Опис профілю цільової групи потенційних клієнтів** | **Готовність споживачів сприйняти продукт** | **Орієнтовний попит в межах цільової групи (сегменту)** | **Інтенсивність конкуренції в сегменті** | **Простота входу у сегмент** |
|---|---|---|---|---|---|
| 1 | Стартапери | Готові | Високий | Середня | Дуже складно |

| | | | | | |
|---|---|---|---|---|---|
| 2 | Державні установи | Потребують переговорів | Середній | Низька | Дуже складно |
| 3 | Ентерпрайз | Потребують переговорів | Середній | Низька | Дуже складно |
| Які цільові групи обрано: державні установи | | | | | |

Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку (таблиця 6.15).

Таблиця 6.15. Визначення базової стратегії розвитку

| **No п/п** | **Обрана альтернатива розвитку проекту** | **Стратегія охоплення ринку** | **Ключові конкурентос-проможні позиції відповідно до обраної альтернативи** | **Базова стратегія розвитку*** |
|---|---|---|---|---|
| 1 | Орієнтація поточної моделі на ринок державних установ | Стратегія концентрованого маркетингу | Державні установи діють за застарілими схемами і тільки в деяких областях з'являється автоматизація процесів | Стратегія диференції |

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів.

Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Після визначення потенційних груп клієнтів проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому впровадженню проекту.

Наступним кроком є вибір стратегії конкурентної поведінки (таблиця 6.16).

Таблиця 6.16. Визначення базової стратегії конкурентної поведінки

| **No п/п** | **Чи є проект «першопрохідцем» на ринку?** | **Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?** | **Чи буде компанія копіювати основні характеристики товару конкурента, і які?** | **Стратегія конкурентної поведінки*** |
|---|---|---|---|---|
| 1 | Так | шукати нових споживачів | Ні | Стратегія заняття конкурентної ніші |

З обраних сегментів до постачальника (державні установи) та до продукту розробляється стратегія позиціонування (таблиця 6.17). що полягає у формуванні ринкової позиції, за яким споживачі мають ідентифікувати торгівельну марку/проект. [**Ошибка! Источник ссылки не найден.**]

Таблиця 6.17. Визначення стратегії позиціонування

| **No п/п** | **Вимоги до товару цільової аудиторії** | **Базова стратегія розвитку** | **Ключові конкуренто-спроможні позиції власного стартап-проекту** | **Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)** |
|---|---|---|---|---|
| 1 | Стабільність роботи<br>Невисока ціна (корпоративна закупівля ліцензії на цілий відділ)<br>Наявність документації<br>Підтримка мобільних платформ | Стратегія диференціації | Державні установи діють за застарілими схемами і тільки в деяких областях з’являється автоматизація процесів | пришвидшення робочих процесів<br>підтримка декількох платформ |

## 6.5. Розроблення маркетингової програми стартап-проекту

Для цього у таблиці 6.18 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару. [**Ошибка! Источник ссылки не найден.**]

Таблиця 6.18. Визначення ключових переваг концепції потенційного товару

| **No п/п** | **Потреба** | **Вигода, яку пропонує товар** | **Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)** |
|---|---|---|---|
| 1 | Пришвидшення робочих процесів | Вузьконаправлена спеціалізація | Більшість конкуренти не мають шаблону для організації процесів |
| 2 | Підтримка мобільних платформ | Підтримка Android, iOS, наявний гнучкий Rest API | Прямий конкурент орієнтований на веб |

Надалі розробляється трирівнева маркетингова модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (таблиця 6.19).

Таблиця 6.19. Опис трьох рівнів моделі товару

| **Рівні товару** | **Сутність та складові** | | |
|---|---|---|---|
| **I. Товар за задумом** | Додаток для організації наукових та освітніх аспектів кафедри | | |
| **II. Товар у реальному виконанні** | Властивості/характеристики | М/Нм | Вр/Тх /Тл/Е/Ор |
| | можливість оптимізації витрат часу | М | Тл |
| | можливість оптимізації витрат коштів | М | Вр |
| | відповідність актуальним технологіям | Нм | Тх |

| | |
|---|---|
| | Відповідає вимогам ДСТУ ISO/IEC 25030:2015 Програмна інженерія. Вимоги щодо якості та оцінювання програмного продукту |
| | Пакування: готовий до використання apk-файл |
| | Марка: К412 |
| **III. Товар із підкрі-пленням** | Потенційний користувач може ознайомитись з поточним товаром з наукових конференцій та публічних виступів, а також наукових вісників на яких була представлена інформація про даний продукт |
| За рахунок чого потенційний товар буде захищено від копіювання: Назва і контент захищені ліцензією; захист інтелектуальної власності | |

М/Нм – монотонні або немонотонні;

Вр/Тх/Тл/Е/Ор – вартісні, технічні, технологічні, ергономічні або органолептичні (останній – для продуктів харчування)

Після формування маркетингової моделі товару слід особливо відмітити – чим саме проект буде захищено від копіювання. Захист може бути організовано за рахунок захисту ідеї товару (захист інтелектуальної власності), або ноу-хау, чи комплексне поєднання властивостей і характеристик, закладене на другому та третьому рівнях товару. [**Ошибка! Источник ссылки не найден.**]

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (таблиця 6.20).

Таблиця 6.20. Визначення меж встановлення ціни

| **No п/п** | **Рівень цін на товари-замінники** | **Рівень цін на товари-аналоги** | **Рівень доходів цільової групи споживачів** | **Верхня та нижня межі встановлення ціни на товар/послугу** |
|---|---|---|---|---|
| 1 | 125…300 грн | 240…400 грн | 12000…42000 грн | 125…240 грн |

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (таблиця 6.21):

— проводити збут власними силами або залучати сторонніх посередників (власна або залучена система збуту);

— вибір та обґрунтування оптимальної глибини каналу збуту;

— вибір та обґрунтування виду посередників.

Таблиця 6.21. Формування системи збуту

| **No п/п** | **Специфіка закупівельної поведінки цільових клієнтів** | **Функції збуту, які має виконувати постачальник товару** | **Глибина каналу збуту** | **Оптимальна система збуту** |
|---|---|---|---|---|
| 1 | Клієнт повинен надаватися в режимах “тріал” та “повний”. Сплата має проходити через AppStore | Легість в встановленні, легкість в сплаті послуг | 4: Розробник даного продукту - Веб-сайт – AppStore- Користувач. | Проводити збут силами посередника AppStore |

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (таблиця 6.22).

Таблиця 6.22. Концепція маркетингових комунікацій

| **No п/п** | **Специфіка поведінки цільових клієнтів** | **Канали комунікацій, якими користуються цільові клієнти** | **Ключові позиції, обрані для позиціонування** | **Завдання рекламного повідомлення** | **Концепція рекламного звернення** |
|---|---|---|---|---|---|
| 1 | Купляють програми через авторизовану мережу AppStore | AppStore, веб-сайти | підтримка декількох платформ Організація робочих процесів | Довести, що програмний продукт полегшить роботу кафедри | Все що треба - тут |

## Висновки до розділу 6

Розроблений програмний продукт має переваги над існуючими конкурентами та є конкурентноздатним на ринку. Програма має шляхи подальшого розвитку, визначені маркетингові стратегії та шляхи збуту. Основна цільова аудиторія – це державні установи, для яких важливо поступово налагодити організацію багатьох процесів, що на даний момент виконуються за застарілими схемами.

# ВИСНОВКИ

Розроблено онлайн-сервіс для організації наукової роботи кафедри.

Було проведено аналіз предметної області та виділено сутності системи, проаналізовано вже існуюче програмне забезпечення, обрано засоби розробки, реалізовано функціонал системи.

В результаті пошуку аналогічних функціональних рішень не було знайдено сервіси, які б покривали функції розробленого програмного продукту.

За допомогою даного програмного продукту можна позбутися певного спектра застарілих організаційних моментів науково-педагогічної діяльності на кафедрі.

Серверний та клієнтський додатки написано у відповідності до сучасних підходів в програмуванні, опираються на технології, що використовуються у багатьох зростаючих проектах.

Завдяки модульній організації розроблене рішення може бути легко розширене додатковим функціоналом, надійне та просте у підтримці, на що також впливає новий «стек» засобів, що зводять програмування до все більш абстрактного рівня.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Sherwin John Calleja Tragura. (2017) Spring 5.0 Cookbook. Packt. p. 670
2. Mick Knutson, Robert Winch, Peter Mularien (2017) Spring Security, 3rd Edition. Packt. p. 542.
3. Arpan Pal. (2017) Iot Technical Challenges and Solutions. Packt. p. 204
4. Matt Neuburg (2018) iOS 12 Programming Fundamentals with Swift: Swift, Xcode, and Cocoa Basics. O'reilly. p.652
5. Russell Fustino (2018) Azure and Xamarin Forms: Cross Platform Mobile Development. apress. p. 260
6. Keith Moon (2017) Swift 4 Programming Cookbook. Packt. p. 384
7. Chris Griffith (2017) Mobile App Development with Ionic 2: Cross-Platform Apps with Ionic 2, Angular 2, and Cordova O'reilly. p. 310.
8. Westley Knight (2018) UX for Developers: How to Integrate User-Centered Design Principles Into Your Day-to-Day Development Work. Apress. p. 166.
9. Muhammad Usama bin Aftab (2017) Building Bluetooth Low Energy Systems. Packt. p. 242
10. Daniel Schwarz (2016) Jump Start Sketch. Sitepoint. p. 150
11. Daniel Schwarz (2017) Jump Start Adobe XD. Sitepoint. p. 170
12. Aanand Shekhar Roy, Rashi Karanpuria. (2018) Kotlin Programming Cookbook. Packt. p. 434
13. Shameer Kunjumohamed. (2016) Spring MVC: Designing Real-World Web Applications. Packt. p. 944
14. Florent Pillet (2017) RxSwift: Reactive Programming with Swift, 2nd Edition. p. 446
15. Apple. (2018). Human Interface Guidelines. p. 388
16. Bogunuva Mohanram Balachandar (2017) RESTful Java Web Services, 3rd Edition. Packt. p. 420

17. Craig Walls. (2016) Spring Boot in Action. Manning publication. p. 264
18. Chris Richardson. (2018) Microservices Patterns: With examples in Java. Manning publication. p. 520
19. Robert C. Martin. (2017) Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice hall. p. 432
20. Marten Deinum. (2018) Spring Boot 2 Recipes: A Problem-Solution Approach. Apress. p. 332.
21. Ottinger, Joseph, Linwood, Jeff, Minter, Dave. (2016) Beginning Hibernate: For Hibernate 5, 4th Edition. Apress. p. 223.
22. Allen G. Taylor. (2018) SQL For Dummies, 9th Edition. p. 512
23. Rivu Chakraborty. (2017) Reactive Programming in Kotlin. Pakt. p.322
24. Ron Ballard. (2017) Relational Databases for Agile Developers. p. 354
25. Laine Campbell, Charity Majors. (2017) Database Reliability Engineering: Designing and Operating Resilient Database Systems. O'reilly. p. 292
26. Ranga Rao Karanam (2018) Spring: Microservices with Spring Boot. Packt. p. 140
27. Krunal Patel, Nilang Patel. (2018) Java 9 Dependency Injection. Packt. p. 246
28. Tomcy John. (2018) Hands-On Spring Security 5 for Reactive Applications. Packt. p. 268
29. Fernando Doglio. (2018) REST API Development with Node.js, 2nd Edition. Packt. p. 323
30. M. Nardone, M. Schincariol, M. Keith. (2018) Pro JPA 2 in Java EE 8: An In-Depth Guide to Java Persistence APIs, 3rd Edition. Apress. p. 759
31. Paul Fisher, Brian D. Murphy. (2016) Spring Persistence with Hibernate, 2nd Edition. Apress. p.184
32. Matt Neuburg. (2018) Programming iOS 12: Dive Deep into Views, View Controllers, and Frameworks. o'reilly. p. 1176
33. Paul Deitel, Harvey Deitel. (2015) Swift for Programmers. Apress. p. 400
34. Jon Hoffman. (2017) Swift 4 Protocol-Oriented Programming, 3rd Edition. Packt. p. 210
35. Brett Ohland, Jayant Varma. (2016) Xcode 7 Essentials, 2nd Edition. Packt. p. 256

36. Robert C. Martin (2008) Clean Code: A Handbook of Agile Software Craftsmanship. Prentice hall. p. 464
37. Emil Atanasov. (2018) Learn Swift by Building Applications: Explore Swift programming through iOS app development. Packt. p. 366
38. Kelvin Lau. (2018) Data Structures & Algorithms in Swift: Implementing practical data structures with Swift 4. Aspress. p. 328
39. Martin Fowler. (2018) Refactoring: Improving the Design of Existing Code, 2nd Edition. Packt. p. 228
40. Erich Gamma. (1994) Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional. p. 395

# ДОДАТОК А

Публікації

iOS додаток управління педагогічними та науковими аспектами роботи кафедри

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТІ31234_18М

Аркушів 9

2018

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»**

# СУЧАСНІ ПРОБЛЕМИ НАУКОВОГО ЗАБЕЗПЕЧЕННЯ ЕНЕРГЕТИКИ

Матеріали XVI Міжнародної науково-практичної конференції аспірантів, магістрантів і студентів м. Київ, 24-27 квітня 2018 року,

ТОМ 2

**Київ- 2018**

УДК 524.36

**Сучасні проблеми наукового забезпечення енергетики:** Матеріали XVI Міжнародної науково-практичної конференції аспірантів, магістрантів і студентів, м. Київ, 24–27 квітня 2018 р. У 2 т. – К. : 7 КПІ ім. Ігоря Сікорського», 2018. – Т. 2. – 298 с.

**ISBN 978-966-622-886-7**
**ISBN 978-966-622-888-1 (Т.2)**

Подано тези доповідей XVI Міжнародної науково-практичної конференції аспірантів, магістрантів і студентів «Сучасні проблеми наукового забезпечення енергетики» за напрямами: автоматизація теплоенергетичних процесів, геометричне моделювання та проблеми візуалізації, програмне забезпечення інформаційних систем та мережних комплексів, моделювання та аналіз теплоенергетичних процесів.



*Друкується в авторській редакції за рішенням Вченої ради теплоенергетичного факультету Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (протокол № 8 від 26 березня 2018 р.)*



**ISBN 978-966-622-886-7**
**ISBN 978-966-622-888-1 (Т.2)**

СЕКЦІЯ №5

# Автоматизація теплоенергетичних процесів

**УДК 004.42**

Магістрант 5 курсу, гр. ТІ-71мп Касьяненко І.І.
Доцент, к.т.н. Карпенко Є.Ю.

## IOS ДОДАТОК WEB-СИСТЕМИ УПРАВЛІННЯ ОРГАНІЗАЦІЙНИМИ АСПЕКТАМИ РОБОТИ КАФЕДРИ

Розвиток комп'ютерів і мережі інтернет створює основу для стрімкого поширення розподілених мобільних та веб додатків. Багато тривіальних розподілених систем застарівають та заміняються більш сучасними та ефективними електронними аналогами.

Особливу увагу слід звернути на системи, які забезпечують навчальний процес.

В розвинених європейських країнах питання модернізації освіти ставиться пріоритетною ціллю.

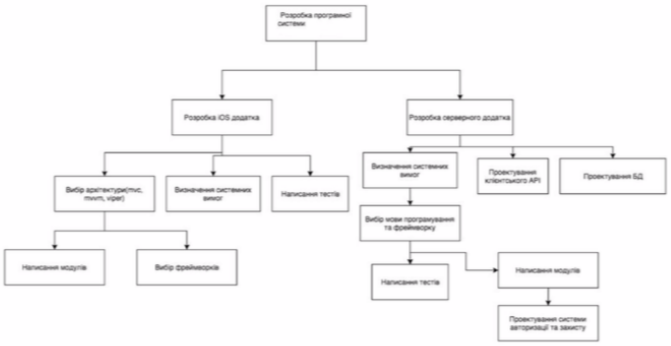
В Україні проблеми освіти розглядаються недостатньо [1]. Нормою є застарілі програми, методології навчання, майже відсутнє впровадження електронних технологій в освіту, це все призводить до загальної регресії освітньої системи.

Однією з насущних проблем вищих навчальних закладів є повільні та малоефективні схеми взаємодії студентів з викладачами, відсутність уніфікованого інструменту впливу на процес навчання через інтернет, застарілість організації процесу вибору тем дипломних робіт студентів та їх узгодження.

Не завжди узгоджений графік прийому та перевірки лабораторних робіт, нечіткі терміни та правила спричиняють створення черг студентів, розпливчате усвідомлення процесу написання та здачі завдань, це все спричиняє неефективний розподіл часу та зусиль студентів та викладачів.

Роздрібненість інформації про кафедри, їх направлення, досягнення створює розпливчасте усвідомлення загального стану факультету, тим самим, відштовхуючи абітурієнтів.

Було запропоновано створити розподілену систему [2], яка дозволить зручно та ефективно організувати процес взаємодії викладачів зі студентами, надасть консолідовану інформацію про напрями та роботу кафедр, дозволить вирішувати організаційні моменти в онлайн режимі.




Перелік посилань:
1. https://mon.gov.ua/ua/tag/yakist-osviti
2. http://www.sm-cloud.com/ios-client-server-integration-approach/

Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут»
Теплоенергетичний факультет

Київська державна академія водного транспорту
імені П.Конашевича-Сагайдачного

Інститут кібернетики ім.В.М.Глушкова НАН

V науково-практична дистанційна конференція
молодих вчених і фахівців
з розробки програмного забезпечення

# «СУЧАСНІ АСПЕКТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

**15 травня 2018**

м. Київ

УДК 004.42:519.7
ББК 973.20-018.2я7

*Рекомендовано Вченою радою Теплоенергетичного факультету*
*Національного технічного університету України*
*«Київський політехнічний інститут»*
*(протокол № від року )*

**Сучасні аспекти розробки програмного забезпечення:** Збірник наукових праць V науково-практичної дистанційної конференції молодих вчених і фахівців з розробки програмного забезпечення, 15 травня 2018 р. – Черкаси: видавець Чабаненко Ю.А., 2018. – 216 с.

У збірнику наукових праць V науково-практичної дистанційної конференції молодих вчених і фахівців з розробки програмного забезпечення «Сучасні аспекти розробки програмного забезпечення» наведено результати наукових досліджень та практичних розробок за наступними напрямами: інженерія програмного забезпечення, комп'ютерний еколого-економічний моніторинг, системи автоматизованого проектування.

*Матеріали друкуються в авторській редакції.*

ISBN

# ДОДАТОК Б

Акт впровадження

iOS додаток управління педагогічними та науковими аспектами роботи кафедри

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТІ31234_18М

Аркушів 2

2018